Approche multi-perspective centrée exigences de composition de services Web

Maha Driss

N° d'ordre : 4475 ANNÉE 2011

UNIVERSITÉ DE RENNES 1

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1
ET DE L'UNIVERSITÉ DE LA MANOUBA

Mention : Informatique

École doctorale Matisse

présentée par

Maha DRISS

Préparée à l'unité de recherche IRISA
Équipe d'accueil: TRISKELL
Composante universitaire: IFSIC
Et
Au laboratoire de recherche RIADI-GDL
Composante universitaire: ENSI

Approche multi-perspective centrée exigences de composition de services Web

Thèse soutenue à Rennes
le 8 Décembre 2011

devant le jury composé de :

Marie-Odile CORDIER
Université de Rennes 1 / *Présidente*
Hanêne BEN ABDALLAH
Université de Sfax / *Rapporteuse*
Flavio OQUENDO
Université de Bretagne Sud / *Rapporteur*
Jean-Marc JÉZÉQUEL
Université de Rennes 1 / *Directeur de thèse*
Henda HAJJAMI BEN GHÉZALA
Université de la Manouba / *Directrice de thèse*
Yassine JAMOUSSI
Université de la Manouba / *Co-directeur de thèse*

Résumé

Le paradigme architecture orientée service AOS est devenu un standard pour la conception et le développement d'applications distribuées à base de services Web. Malgré tous les avantages qu'elles apportent en termes d'interopérabilité et de réutilisation, les solutions de développement associées au paradigme AOS sont destinées aux programmeurs et restent difficiles à comprendre par le monde de l'entreprise. Pour être en phase avec le monde de l'entreprise, les applications à base de services Web doivent être décrites en termes d'exigences qu'elles permettent de satisfaire et non pas en termes de fonctionnalités qu'elles permettent de réaliser. Ceci permet de minimiser la discordance conceptuelle entre les services logiciels et l'énoncé des exigences des utilisateurs.
Nous proposons dans le cadre de ce travail de thèse, une approche multi-perspective centrée exigences pour la composition de services Web. Notre approche positionne la composition des services dans une perspective centrée exigences dans laquelle les services métiers de haut niveau sont décrits en termes d'exigences qu'ils permettent de satisfaire. Un processus d'alignement est proposé pour assurer une mise en correspondance de ces services avec les services logiciels de bas niveau qui sont décrits en termes de déclarations techniques au niveau d'une perspective centrée fonctions. Ce processus intègre la variabilité de composition tout au long des étapes de construction de ces applications à base de services.

Mots-clés : Services Web, Composition de services, Exigences, Multi-perspective, Alignement, Variabilité.

Abstract

The service-oriented architecture paradigm SOA has become the standard for the design and development of distributed Web service-based-applications. Despite all the advantages they offer in terms of interoperability and reuse, the development solutions associated with the SOA paradigm is intended for programmers and are difficult to understand by the business world. To be in tune with the business world, service-based applications should be described in terms of requirements they can meet and not in terms of features they can achieve. This minimizes the discrepancy between the conceptual software services and the statement of user requirements.
We propose, as part of this thesis, a multi-perspective requirement centric approach for Web service composition. Our approach places the composition of services at a requirement-centered perspective in which high level business services are described in terms of requirements they can meet. An alignment process is proposed to ensure a matching between these services and the low-level software services that are described in terms of technical reports at a function-centered perspective. This process incorporates the composition variability throughout the different phases of the service-based applications development cycle.

Keywords : Web services, Service composition, Requirements, Multi-perspective, Alignement, Variability.

Je dédie cette thèse
à mon cher mari Wadii,
à mes chers parents Noureddine et Zeineb,
à mes soeurs Sawssen et Kaouthar,
à toute ma famille,
à tous mes amis
et à tous ceux qui m'on aidé de près ou de loin...

Remerciements

Au terme de cette expérience riche et passionnante, j'exprime tout d'abord ma gratitude à Madame Marie-Odile CORDIER, Professeure à l'Université de Rennes 1, pour avoir accepté de présider ce jury de thèse.

Je remercie Madame Hanêne BEN ABDALLAH, Professeure à l'Université de Sfax, et Monsieur Flavio OQUENDO, Professeur à l'Université de Bretagne Sud, d'avoir accepté la fastidieuse tâche de rapporter ce travail.

Je tiens à exprimer ma haute reconnaissance à mes directeurs de thèse : Madame Henda HAJJAMI BEN GHÉZALA, Professeure à l'Université de la Manouba, et Monsieur Jean-Marc JÉZÉQUEL, Professeur à l'Université de Rennes 1, pour leur soutien, leur disponibilité et leurs encouragements.

Je suis sincèrement reconnaissante à Monsieur Yassine JAMOUSSI, Maître de conférences à l'Université de la Manouba, pour sa disponibilité, ses précieux conseils et son investissement dans ce travail de recherche.

Mes remerciements vont également à tous les membres de l'équipe TRISKELL et du laboratoire RIADI-GDL et à tous ceux qui m'ont aidé et encouragé tout au long de cette thèse.

Préface

Cette thèse a été réalisée dans le cadre d'une cotutelle entre l'Université de la Manouba et l'Université de Rennes 1. Dans le cadre de cette coopération, Maha DRISS a effectué des séjours alternés à raison en moyenne d'une session (4 mois) en France et de deux sessions (8 mois) en Tunisie sur les 4 ans.

Maha a été accueillie au sein du laboratoire RIADI-GDL dans l'Ecole Nationale des Sciences de l'Informatique (ENSI) à l'Université de la Manouba ainsi qu'au sein de l'équipe projet TRISKELL, INRIA RENNES - BRETAGNE ATLANTIQUE dans l'Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA) à l'Université de Rennes 1.

Table des matières

Liste des figures

Liste des tableaux

Première partie

Problématique et état de l'art

Chapitre 1

Introduction

Ce chapitre décrit le contexte général de nos travaux de thèse qui portent sur le domaine de l'ingénierie des applications à base de services Web. Nous expliquons notre motivation en montrant les atouts d'une composition de services centrée exigences. Nous présentons les problèmes associés à la composition centrée exigences à savoir : la modélisation des exigences, la découverte, la sélection et la coordination des services Web. Enfin, nous présentons le sujet de notre thèse et nos contributions principales qui s'articulent autour de la définition d'une nouvelle approche multi-perspective centrée exigences pour la composition de services Web.

Sommaire

1.1 Contexte : Ingénierie des applications à base de services Web

Le contexte général de la thèse porte sur le domaine de l'ingénierie des applications à base de services Web.

La généralisation de l'usage du Web et l'épanouissement de la technologie client-serveur constituent la nouvelle étape dans la chaîne évolutive de mise en œuvre des applications distribuées qui a mené à l'émergence du paradigme service Web. Les services Web sont apparus comme une nouvelle technologie qui, grâce aux possibilités d'interopérabilité qu'elle offre, se place aujourd'hui comme le point de convergence technologique d'acteurs de divers domaines : e-commerce [1] , e-learning [2], e-government [3], etc. Le W3C [4] définit les services Web comme des composants logiciels offrant une ou plusieurs opérations allant des plus simples aux plus complexes. Ces composants sont publiés, découverts et invoqués à travers le Web grâce à l'utilisation d'Internet comme infrastructure de communication et d'XML comme format de données [96].

L'émergence du paradigme service Web a marqué une évolution significative dans l'histoire d'Internet qui a été destiné à jouer le rôle d'un vecteur d'échange de données. Avec les services Web, Internet se transforme en une plate-forme de composants logiciels auto-descriptifs, facilement intégrables et faiblement couplés [66]. Les services Web sont venus pallier les problèmes qu'ont rencontrés les entreprises en termes d'interopérabilité, et ceci en mettant en place une Architecture Orientée Services (AOS) [31] reposant sur un ensemble de standards. Le processus de standardisation touche trois couches de l'infrastructure de base de l'AOS : le protocole de communication qui permet de structurer les messages échangés entre les services, la spécification de description des interfaces des services et enfin la spécification de publication et de localisation de services.

L'un des avantages les plus importants du paradigme service Web est la réutilisation. Les services Web, tels qu'ils sont présentés, sont conceptuellement limités à des fonctionnalités relativement simples qui sont modélisées par une collection d'opérations. Toutefois, il est nécessaire de construire de nouvelles applications par composition de

1. Méthode d'apprentissage qui repose sur la mise à disposition de contenus pédagogiques en utilisant des nouvelles technologies multimédia et de l'Internet.

2. Processus d'achat et de vente ou d'échange de biens, services et informations, via des réseaux informatiques, notamment Internet.

3. Utilisation des technologies de l'information et de la communication par les administrations publiques pour rendre les services publics plus accessibles à leurs usagers et pour améliorer leur fonctionnement interne.

4. World Wide Web Consortium est un consortium chargé de promouvoir la compatibilité des technologies du World Wide Web.

services afin de répondre à des exigences plus complexes [29]. La tendance ultime de cette nouvelle approche est de mettre à profit les composants services Web au service de l'intégration d'applications en interne ou en externe. C'est ce que nous appelons composition ou agrégation de services Web.

1.2 Motivation : Composition centrée exigences des services Web

L'acceptation et la popularité croissantes de l'architecture AOS ont donné lieu à un ensemble d'opportunités. Dans le cadre de ce travail de thèse, la motivation qui anime notre intérêt pour l'architecture AOS est le développement d'applications distribuées par composition de services Web. La composition de services est une opportunité séduisante puisqu'elle offre des avantages compétitifs aux entreprises en leur offrant la possibilité de développer des applications à valeur ajoutée en assemblant des services déjà existants.

De nombreux travaux de comités académiques et industriels ont abordé le problème de composition de services Web en proposant des langages (par exemple BPEL [64], WS-CDL [63] et OWL-S [61]) et des formalismes (les réseaux de Pétri [59], les machines de Mealy [9] et les machines d'états [34]) divers. Il est toutefois constant de remarquer que tous ces travaux adoptent une vision "centrée fonctions" de la composition en considérant les services comme des composant "logiciels". L'inconvénient de cette vision est qu'elle est dédiée aux développeurs d'applications puisqu'elle se base sur des descriptions de "bas niveau" qui se concentrent sur des déclarations purement techniques (comme les formats des messages de coordination, les types des paramètres d'entrée/sortie et le protocole Web utilisé pour la communication). Toutefois, pour que l'architecture AOS soit transposée au niveau de l'entreprise, il est nécessaire qu'une composition de services soit décrite en termes d'exigences que l'entreprise souhaite atteindre [49]. Ainsi, pour être en phase avec le monde de l'entreprise, il est nécessaire d'opter pour un changement d'échelle d'observation et adopter une vision "centrée exigences" de la composition où les services sont considérés comme des interactions "métiers". Les descriptions des services sont exprimées en termes d'exigences et non plus en termes de fonctionnalités et ceci garanti un "haut niveau" d'abstraction qui renforce la variabilité et la réutilisation. En effet, ces descriptions comportent des variations d'usage et peuvent ainsi être réutilisés dans plusieurs autres compositions.

La vision centrée exigences de la composition de services a fait l'objet de plusieurs travaux de recherche que nous pouvons classifier en trois catégories. Nous trouvons des travaux qui ont opté pour : (i) une approche formelle pour l'expression des exigences [53], [93] et [83], (ii) une approche semi-formelle comme [1], [22], [80] et [81], et finalement (iii) une approche informelle comme [102] et [19]. Ces travaux s'intéressent à proposer des solutions pour la découverte et la composition de services logiciels à

partir des définitions d'exigences. Néanmoins, les solutions qu'ils proposent souffrent de quelques limitations que nous détaillons dans ce qui suit.

1.3 Problématique : Absence d'alignement entre les services métiers et les services logiciels dans le cadre de la composition de services Web

Suite à notre étude des travaux précédents de la littérature, nous avons pu observer que les services métiers et les services logiciels sont traités d'une façon isolée dans le cadre de la composition. En effet, les services métiers ne peuvent être mis en correspondance avec les services logiciels qui les réalisent directement et automatiquement. Ceci peut entraîner des discordances conceptuelles entre l'énoncé des exigences décrivant les services métiers et des fonctionnalités définissant les services logiciels.

Dans le cadre de ce travail de thèse, nous répondons à ce problème principal concernant **l'absence d'alignement entre les services métiers et les services logiciels dans le cadre de la composition de services Web** et nous comblons les limitations listées ci-dessous.

- **Limitation 1. Traitement partiel de la variabilité.** Le besoin de variabilité dans le cadre des applications logicielles a émergé à la suite d'un changement de comportement des utilisateurs qui ne veulent plus s'adapter aux logiciels mais qui préfèrent que ces derniers soient configurés selon leurs besoins [8]. Cet état de fait a forcé les développeurs à prendre en compte les exigences des utilisateurs. Leur objectif n'est plus réduit à l'augmentation de l'efficacité d'un logiciel mais à la proposition d'une solution générique qui couvre le plus grand nombre des besoins. Ainsi, la variabilité est devenue un facteur important dans le développement des applications logicielles et plus particulièrement des applications à base de services Web. La classification proposée par [42] est basée sur la distinction entre deux catégories de variabilité : la variabilité essentielle et la variabilité technique. La variabilité essentielle représente le point de vue de l'utilisateur et s'intéresse aux aspects liés à l'utilisation de l'application tandis que la variabilité technique représente le point de vue du développeur et s'intéresse aux aspects liés à la réalisation de l'application. Suite à notre étude de la littérature qui relate au domaine de la composition de services Web, nous avons pu constater que les deux catégories de la variabilité ont été traitées d'une façon isolée et que la variabilité technique a suscité plus d'intérêt que la variabilité essentielle. En effet, nous trouvons que quelques travaux qui se sont focalisés sur l'expression et la gestion de la variabilité essentielle [1] [80], [81] et [19]. Cependant, nous pensons qu'il est important de tenir compte des deux catégories de la variabilité afin de garder une traçabilité entre les services métiers et les services logiciels qui les réalisent. Cette traçabilité évite d'avoir des compositions de services rigides

et inadaptées aux exigences des utilisateurs.

- **Limitation 2. Négligence des exigences non-fonctionnelles.** Les exigences non-fonctionnelles dans le domaine des services Web sont exprimées par le terme Qualité de Service QdS qui réfère à diverses propriétés telles que la disponibilité, le temps de réponse, la sécurité et le débit [2]. La plupart des travaux comme [80], [102] et [19] se sont concentrés sur l'expression des exigences fonctionnelles et ont négligé les exigences non-fonctionnelles. Toutefois, nous considérons qu'il est nécessaire de tenir compte des exigences non-fonctionnelles qui contraignent la manière dont l'application à base de services doit satisfaire les exigences fonctionnelles. En effet, devant l'accroissement incessant du nombre de services Web disponibles et la variabilité des besoins des utilisateurs, l'opération de composition peut engendrer différents services composites qui offrent les mêmes fonctionnalités. Donc, c'est à la base d'un certain nombre de propriétés de QdS que nous pouvons choisir un service composite et pas un autre.

- **Limitation 3. Pas de découverte et sélection des services logiciels.** La découverte et la sélection des services présentent un processus qui consiste à chercher dans l'annuaire les services logiciels qui répondent le mieux aux exigences fonctionnelles et non-fonctionnelles des utilisateurs. La plupart des travaux comme [93], [83], [1], [22], [80]et [81] proposent d'implémenter les services logiciels qui réalisent les exigences des utilisateurs. Ceci est en contradiction avec l'objectif principal de l'architecture AOS qui vise à réutiliser les services déjà existants pour le développement de nouvelles applications.

- **Limitation 4. Génération en boîte noire du procédé de coordination des services logiciels.** La composition de services est mise en œuvre moyennant un procédé de coordination qui permet de spécifier quels services doivent être invoqués, dans quel ordre et sous quelles pré-conditions. Deux types de coordination de services ont été proposées : une coordination centralisée appelée orchestration et une coordination décentralisée appelée chorégraphie [30]. Toutes des approches qui ont opté pour la vision centrée exigences pour la composition de services n'explicitent pas le passage du modèle des exigences au procédé de coordination des services logiciels qui réalisent ces exigences. La façon dont les services logiciels sont coordonnés et exécutés n'est pas claire et ce manque de transparence empêche toute comparaison ou amélioration des techniques d'alignement proposées.

- **Limitation 5. Validation partielle de la composition.** La validation de la composition consiste à vérifier que les services composés répondent aux exigences fonctionnelles et non-fonctionnelles des utilisateurs. Dans la littérature, nous trouvons des travaux qui se sont focalisés sur la validation des exigences fonctionnelles comme [102] et d'autres qui se sont concentrés sur la validation des exigences non-fonctionnelles comme [83]. Peu sont les travaux qui ont présenté

des résultats de validation portant à la fois sur les exigences fonctionnelles non-fonctionnelles. Cette validation incomplète rend difficile la comparaison entre ces différents travaux et le repérage des points de force et de faiblesse de chacun d'eux.

1.4 Thèse : Une approche multi-perspective centrée exigences pour la composition de services Web

Notre thèse répond aux limitations énoncées ci-dessus en proposant une nouvelle approche multi-perspective centrée exigences pour la composition de services Web. Notre approche positionne la composition des services dans une perspective centrée exigences dans laquelle les services métiers de haut niveau sont décrits en termes d'exigences qu'ils permettent de satisfaire. Un processus d'alignement est proposé pour assurer une mise en correspondance de ces services avec les services logiciels de bas niveau qui sont décrits en termes de déclarations techniques au niveau d'une perspective centrée fonctions.

L'approche que nous proposons dans le cadre de cette thèse permet : (i) la modélisation des applications à base de services en termes d'exigences fonctionnelles et non fonctionnelles ; (ii) la découverte des services logiciels pertinents qui répondent le mieux aux exigences fonctionnelles ; (ii) la sélection des services logiciels pertinents et de haute QdS ; (iv) la coordination des services logiciels sélectionnés en vue de les orchestrer ; (v) la validation de la composition afin de vérifier qu'elle répond aux exigences des utilisateurs.

Notre approche consiste donc en un processus découpé en cinq étapes successives 1-5 comme le montre la Figure 1.1.

- **Étape 1. Modélisation centrée exigences des services métiers.** À cette étape, nous proposons une modélisation centrée exigences des services métiers. Cette modélisation consiste à représenter les exigences des utilisateurs et à identifier et spécifier les services métiers et leur composition à partir de ces exigences.

- **Étape 2. Découverte des services logiciels pertinents.** À cette étape, nous proposons de chercher les services logiciels pertinents qui répondent le mieux aux exigences fonctionnelles. La recherche est basée sur les spécifications des services métiers élaborées dans la première étape. Différents niveaux de filtrage sont définis et appliqués pour réduire l'ensemble des services logiciels trouvés et pour faciliter la sélection dans l'étape suivante.

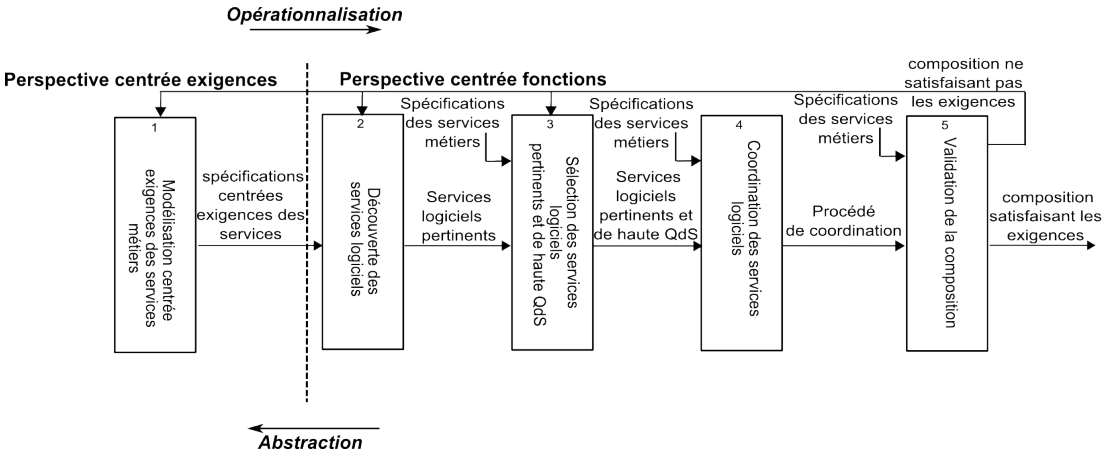


Figure 1.1 — Approche multi-perspective centrée exigences pour la composition de services Web.

- **Étape 3. Sélection des services logiciels pertinents et de haute QdS.** À cette étape, nous proposons une méthode de guidage automatique pour la sélection des services logiciels pertinents et de haute QdS.

- **Étape 4. Coordination des services logiciels.** À cette étape, nous proposons une génération automatique du procédé de coordination à partir du modèle des exigences élaboré dans la première étape.

- **Étape 5. Validation de la composition.** À cette étape, nous proposons une validation empirique des exigences fonctionnelles et une validation automatique des exigences non-fonctionnelles.

Des boucles de retour existent entre les étapes dans le cas où les résultats de validation obtenus montrent que les services composés ne répondent pas aux exigences des utilisateurs. Dans ce cas, nous proposons des affinements au niveau des trois premières étapes : (i) Modélisation centrée exigences des services métiers : le modèle et les spécifications des services métiers peuvent être affinés en modifiant des exigences fonctionnelles et/ou des contraintes de QdS, (ii) Découverte des services logiciels pertinents : d'autres mots-clés peuvent être utilisés pour chercher un service logiciel, ou (iii) Sélection des services logiciels pertinents et de haute QdS : d'autres services peuvent être sélectionnés à partir de l'ensemble des services logiciels retourné par l'étape de découverte.

Ce processus est itératif et il peut être exécuté plusieurs fois jusqu'à ce que les résultats de validation montrent que la composition obtenue satisfait au mieux les exigences fonctionnelles et non-fonctionnelles des utilisateurs.

1.5 Contributions

Les travaux entrepris dans le cadre de cette thèse ont menés aux contributions suivantes :

- **Contribution 1. La définition et la caractérisation de deux perspectives pour la composition de services Web.** *En réponse à la limitation 1. Traitement partiel de la variabilité.* Nous définissons deux perspectives pour la composition de services : une perspective centrée exigences appelée perspective "intentionnelle" qui intègre la variabilité essentielle dans le modèle de spécification des services métiers et une perspective centrée fonctions appelée perspective "opérationnelle" qui intègre la variabilité technique dans le modèle de coordination et d'exécution des services logiciels.

- **Contribution 2. La proposition d'une couche de QdS.** *En réponse à la limitation 2. Négligence des exigences non-fonctionnelles.* Nous ajoutons une couche QdS au modèle utilisé pour la spécification des services métiers.

- **Contribution 3. L'extension d'un outil pour la découverte automatique des services logiciels pertinents.** *En réponse à la limitation 3. Pas de découverte et sélection des services logiciels.* Nous procédons à la découverte des services logiciels par l'utilisation et l'extension d'un moteur de recherche de services. La découverte est réalisée en interrogeant le moteur de recherche avec un ou plusieurs mots-clés extraits à partir des spécifications des services métiers. Pour réduire l'ensemble des services logiciels retourné par le moteur de recherche et pour faciliter la sélection par la suite, nous définissons et nous appliquons différents niveaux de filtrage (de QdS, syntaxique, sémantique, etc.) pour en garder que les services les plus pertinents qui répondent le mieux aux exigences fonctionnelles.

- **Contribution 4. La définition et la mise en œuvre d'une méthode de guidage automatique pour la sélection des services logiciels pertinents et de haute QdS.** *En réponse à la limitation 3. Pas de découverte et sélection des services logiciels.* Nous définissons et nous mettons en œuvre une méthode de guidage automatique pour la sélection des services logiciels pertinents et de haute QdS. Cette méthode est basée sur deux cadres de travail formels qui sont l'Analyse Formelle de Concepts (AFC) et l'Analyse Relationnelle de Concepts (ARC) et elle s'applique aussi bien pour la sélection des services logiciels atomiques que des services logiciels composites.

- **Contribution 5. L'implémentation d'un mécanisme de génération automatique des procédés de coordination des services logiciels.** *En réponse à la limitation 4. Génération en boîte noire du procédé de coordination des services logiciels.* Nous implémentons une transformation de modèles qui

permet de générer automatiquement les procédés de coordination des services logiciels à partir des spécifications des services métiers.

– **Contribution 6. La Validation de la composition.** *En réponse à la limitation 4. Validation partielle de la composition.* Nous validons empiriquement en termes de précision et de rappel la composition des services par rapport aux exigences fonctionnelles. La précision évalue le nombre de réels services pertinents et de haute QdS parmi les services sélectionnés et composés par notre approche, tandis que le rappel évalue le nombre de services sélectionnés et composés par notre approche parmi les réels services pertinents et de haute QdS qui existent. Pour les exigences non-fonctionnelles, nous proposons d'appliquer la technique de simulation pour vérifier que les services sélectionnés et composés répondent au mieux à ces exigences.

1.6 Plan

Ce mémoire de thèse est découpé en trois grandes parties qui sont organisées comme suit :

– **La première partie :** inclut le chapitre 1 d'introduction déjà présenté, le chapitre 2 qui donne un aperçu sur les concepts de base du paradigme services Web et le chapitre 3 qui présente un état de l'art sur les approches d'ingénierie des exigences qui traitent les applications à base de services Web. Ce troisième chapitre se termine par une mise en perspective de notre approche avec les travaux de la littérature.

– **La deuxième partie :** inclut le chapitre 4 qui expose l'approche proposée : l'approche multi-perspective centrée exigences de composition de services Web et le chapitre 5 qui illustre cette approche par une étude de cas.

– **La troisième partie :** conclut ce mémoire de thèse en récapitulant les contributions apportées dans le chapitre 6 et en annonçant les perspectives de travaux de recherche futurs dans le chapitre 7.

Chapitre 2

Concepts de base du paradigme service Web

Ce chapitre établit une étude du fondement théorique de notre travail de thèse à savoir les concepts de base du paradigme service Web. Nous commençons d'abord par présenter un tour d'horizon définissant l'infrastructure et l'architecture de référence de ce nouveau paradigme. Ensuite, nous nous s'intéressons à élucider la notion de composition de services en introduisant ses différents types ainsi que les procédés utilisés pour décrire la coordination.

Sommaire

2.1 Les services Web : Vue d'ensemble

L'une des tendances historiques qui a conduit à l'apparition des services Web est l'utilisation de l'architecture par composants. Ce type d'architecture a engendré un développement rapide et évolutif d'applications distribuées et complexes. Au cours de ce développement, nous avons assisté à la mise en place de trois technologies de distribution [30] : CCM [1](CORBA Component Model), EJB [2] (Enterprise JavaBeans) et .Net [3]. La mise en œuvre de ces trois technologies a soulevé de nombreuses difficultés [88] :

- **Difficulté 1. L'interdépendance des composants.** Pour préserver son évolutivité, une application distribuée doit minimiser l'interdépendance entre ses composants. Cependant, cette action peut entraîner un dysfonctionnement dont il est souvent difficile de déterminer précisément l'origine.

- **Difficulté 2. Hétérogénéité des formats des composants.** Pour préserver sa qualité de service, une application distribuée doit garantir une interopérabilité entre ses divers composants et une maîtrise de leur hétérogénéité. Cependant, la distribution de composant est un enjeu industriel où chaque technologie veut imposer son format, bien évidemment, incompatible avec les autres formats.

Ces difficultés ont contribué à l'apparition d'une nouvelle architecture de distribution où les composants sont indépendants, autonomes et décrits sous un seul format standardisé. Cette architecture est l'architecture orientée services [88].

2.1.1 Définition

Les Web services constituent une approche pour mettre en œuvre le paradigme de service. Les Web services ont été proposés initialement par IBM et Microsoft, puis en partie standardisés par le consortium du World Wide Web : le W3C.

Selon W3C [96] : *"Un service Web est un système conçu pour permettre l'interopérabilité des applications à travers un réseau. Il est caractérisé par un format de description interprétable/compréhensible automatiquement par la machine (en particulier WSDL). D'autres systèmes peuvent interagir avec le service Web selon la*

1. Architecture proposée par l'OMG (Object Management Group) permettant la définition de composants portables à l'aide de l'IDL (Interface Definition Language) et leur interopérabilité à l'aide du bus applicatif ORB (Object Request Broker) et du protocol IIOP (Internet Inter-ORB Protocol).

2. Architecture développée par Sun proposant un cadre pour créer des composants distribués écrits en langage Java et exécutés sur la plateforme J2EE (Java 2 Entreprise Edition).

3. Cadre de travail introduit par Microsoft proposant une approche unifiée pour le développement d'applications Windows ou Web par intégration de composants. Il s'appuie sur la norme CLI (Common Language Infrastructure) qui est indépendante du langage de programmation utilisé pour les composants.

manière prescrite dans sa description et en utilisant des messages SOAP, généralement transmis via le protocole HTTP et sérialisés en XML et en d'autres standards du Web".

Selon M. P. Papazoglou [66] : *"Les services Web sont des éléments auto-descriptifs et indépendants des plateformes qui permettent la composition à faible coût d'applications distribuées. Les services Web effectuent des fonctions allant de simples requêtes à des processus métiers complexes. Les services Web permettent aux organisations d'exposer leurs programmes résultats sur Internet (ou sur un intranet) en utilisant des langages (basés sur XML) et des protocoles standardisés et de les mettre en œuvre via une interface auto-descriptive basée sur des formats standardisés et ouverts".*

Techniquement, un service Web est donc une entité logicielle offrant une ou plusieurs fonctionnalités allant des plus simples aux plus complexes. Ces entités sont publiées, découvertes et invoquées à travers le Web grâce à l'utilisation d'Internet comme infrastructure de communication ainsi que de formats de données standardisés comme XML.

2.1.2 Objectifs

L'approche de service Web vise essentiellement quatre objectifs fondamentaux expliquant son grand succès [46] [31] :

- **L'interopérabilité** : l'interopérabilité permet à des applications écrites dans des langages de programmation différents et s'exécutant sur des plateformes différentes de communiquer entre elles. En manipulant différents standards que ce soit XML ou les protocoles d'Internet, les services Web garantissent un haut niveau d'interopérabilité des applications et ceci indépendamment des plateformes sur lesquelles elles sont déployées et des langages de programmation dans lesquels elles sont écrites. Ainsi, en s'appuyant sur un format d'échange de messages standard et sur l'ubiquité de l'infrastructure d'Internet, l'interopérabilité est donc une caractéristique intrinsèque aux services Web.

- **Le couplage faible** : Le couplage est une métrique indiquant le niveau d'interaction entre deux ou plusieurs composants logiciels. Deux composants sont dits couplés s'ils échangent de l'information. Nous parlons de couplage fort si les composants échangent beaucoup d'information et de couplage faible dans le cas contraire. Vu que la communication avec les services Web est réalisée via des messages décrits par le standard XML caractérisé par sa généricité et son haut niveau d'abstraction, les services Web permettent la coopération d'applications tout en garantissant un faible taux de couplage. Par conséquent, il est possible de modifier un service sans briser sa compatibilité avec les autres services composant l'application.

– **La réutilisation** : L'avantage de la réutilisation est qu'elle permet de réduire les coûts de développement en réutilisant des composants déjà existants. Dans le cas de l'approche service Web, l'objectif de la séparation des opérations en services autonomes est en effet pour promouvoir leur réutilisation. Ainsi, lorsqu'un client définit ses exigences, il est généralement possible de réutiliser des services déjà existants pour satisfaire une partie des exigences. Ceci facilite la maintenance de l'application et permet un gain de temps considérable.

– **La découverte et la composition automatiques** : La découverte et la composition sont des étapes importantes qui permettent la réutilisation des services. En effet, il faudra être en mesure de trouver et de composer un service afin de pouvoir en faire usage. En exploitant les technologies offertes par Internet et en utilisant un ensemble de standards pour la publication, la recherche et la composition, l'approche services Web tend à diminuer autant que possible l'intervention humaine en vue de permettre une découverte et une composition automatiques des services les plus complexes. En effet, pour réaliser son application, un développeur peut simplement interroger un moteur de recherche de services afin de trouver le service adéquat et à l'aide de langages de coordination appropriés il peut l'intégrer avec le reste des services de son application.

2.1.3 Exemple de socle technologique : le trio SOAP, WSDL et UDDI

Les services Web sont construits autour de standards qui sont SOAP, WSDL et UDDI assurant respectivement leur transport, leur description et leur découverte.

2.1.3.1 Transport : SOAP

SOAP (Simple Object Access Protocol) est un protocole spécifié par le W3C [97]. Il assure la communication entre clients et services ou entre services par échange de messages au travers du Web. Il utilise principalement les protocoles HTTP [4] (Hyper-Text Tranfer Protocol) et SMTP [5] (Simple Mail Transfer Protocol) pour le transport de messages. Contrairement aux protocoles utilisés pour les autres technologies de distribution comme IIOP [6] (Internet Inter-ORB Protocol) pour CCM ou JRMP [7] (Java Remote Method Protocol) pour EJB qui sont des protocoles binaires, SOAP se base sur le standard XML pour encoder les données.Par conséquent, il profite des avantages de généricité, d'abstraction et de portabilité qu'offre ce standard pour la

4. Protocole de communication client-serveur développé par le World Wide Web. Il permet l'échange de tout type de données entre un client et un serveur en utilisant le port 80.

5. Protocole de communication utilisé pour transférer le courrier électronique vers les serveurs de messagerie électronique moyennant le port 25.

6. Protocole de communication utilisé par CORBA. C'est une implémentation s'appuyant sur un transport TCP/IP du protocole de plus haut-niveau GIOP (General Inter-ORB Protocol).

7. Protocole de communication utilisé par l'API RMI (Remote Method Invocation). Il assure les connexions et les transferts de données et ceci en utilisant le langage Java et le protocole TCP/IP.

normalisation et la structuration des données. Les messages SOAP sont englobés dans une enveloppe constituée d'un entête et d'un corps. L'entête est facultatif et il apporte des données supplémentaires au message SOAP comme des informations concernant l'authentification, la gestion de transactions, le paiement, etc. Le corps renferme, du côté client, l'opération du service invoquée ainsi que des valeurs des paramètres nécessaires à cette invocation, et du côté service, le résultat de l'exécution de l'opération invoquée.

La Figure 2.1 montre un message requête SOAP vers le service *HelloWorld* pour appeler l'opération *makeHelloWorld* prenant un seul paramètre de type chaîne de caractères value et le message de retour SOAP de l'appel de cette opération qui retourne aussi un seul paramètre de type chaîne de caractères *helloWorldResult*.

Requête vers le service HelloWorld

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Body>
<makeHelloWorld xmlns="http://makeHelloWorld">
<value>Maha DRISS</value>
</makeHelloWorld>
</soapenv:Body>
</soapenv:Envelope>
```

Réponse du service HelloWorld

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Body>
<makeHelloWorldResponse xmlns="http://makeHelloWorldResponse">
<helloWorldResult>Hello World to Maha DRISS</helloWorldResult>
</makeHelloWorldResponse>
</soapenv:Body>
</soapenv:Envelope>
```

Figure 2.1 — Exemple de message requête SOAP appelant l'opération *makeHelloWorld* du service HelloWorld et son message réponse SOAP [21].

2.1.3.2 Description : WSDL

WSDL (Web Service Description Language) est un langage de description de services Web proposé par le W3C [98]. Il est basé aussi sur le standard XML. Il renseigne sur les aspects techniques relatifs à l'implémentation et à l'invocation du service Web à savoir son adresse, le protocole qu'il utilise, les opérations qu'il offre et les types des données échangées.

Un document WSDL est décomposé en deux parties : une partie abstraite et une partie concrète. La partie abstraite décrit les messages et les opérations disponibles via les éléments suivants :

- <**types**> (optionnel et un seul autorisé) : informe sur les structures de données des paramètres utilisés par le service.

- <**message**> (plusieurs autorisés) : encapsule les données véhiculées pour l'opération invoquée. Pour chaque opération du service, il existe deux éléments <**message**> correspondant respectivement à la requête et à la réponse.

- <**portType**> (plusieurs autorisés) : composé d'une ou plusieurs opérations décrites par des éléments <**operation**>. Chaque opération possède un nom, 0 ou plusieurs messages en entrée et 0 ou plusieurs messages de sortie ou de fautes.

La partie concrète décrit le protocole et le type d'encodage à utiliser pour les messages Elle est composée par les éléments suivants :

- <**binding**> (plusieurs autorisés) : définit les protocoles de communication utilisés pour l'invocation du service Web. Cette définition permet d'établir le lien, d'une part, entre le document et les messages SOAP et d'autre part, entre les messages SOAP et les opérations invoquées.

- <**service**> (plusieurs autorisés) : permet de décrire un service comme étant un ensemble de ports à travers lesquels il est possible d'accéder à ses opérations. Ces ports représentent les adresses URI (Uniform Resource Identifier) du service.

Plusieurs parties concrètes peuvent être proposées pour une même partie abstraite. Cette séparation des éléments du document WSDL en deux parties renforce la réutilisabilité de la partie abstraite.
La Figure 2.2 présente le document WSDL du service *HelloWorld*. Le service offre deux opérations : la première opération est l'opération *makeHelloWorld* qui prend en paramètre une chaîne de caractères et retourne une chaîne caractères et la deuxième opération est l'opération *simpleHelloWorld* sans paramètre en entrée et qui retourne une chaîne de caractères. L'accès au service est réalisé par l'intermédiaire de messages SOAP. Le protocole utilisé pour l'échange des messages SOAP est HTTP et le style est RPC. Le service *HelloWorld* est accessible à partir de son adresse URI : *http ://hello-worldwebservice*.

2.1.3.3 Découverte : UDDI

UDDI (Universal Description, Discovery and Integration) est une standardisation pour la publication et la découverte des services Web [62]. UDDI est sponsorisée par

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<definitions name="HelloWorld"
targetNamespace="http://helloworldwebservice"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:tns="http://helloworldwebservice"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
<types/>
<message name="makeHelloWorld">
<part name="value" type="xsd:string"/>
</message>
<message name="makeHelloWorldResponse">
<part name="helloWorldResult" type="xsd:string"/>
</message>
<message name="simpleHelloWorld"/>
<message name="simpleHelloWorldResponse">
<part name="helloWorldResult" type="xsd:string"/>
</message>
<portType name="HelloWorld">                    Partie abstraite
<operation name="makeHelloWorld">
<input message="tns:makeHelloWorld"/>
<output message="tns:makeHelloWorldResponse"/>
</operation>
<operation name="simpleHelloWorld">
<input message="tns:simpleHelloWorld"/>
<output message="tns:simpleHelloWorldResponse"/>
</operation>
</portType>
<binding name="HelloWorldPortBinding" type="tns:HelloWorld">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
<operation name="makeHelloWorld">
<soap:operation soapAction=""/>
<input>
<soap:body use="literal" namespace="http://helloworldwebservice"/>
</input>
<output>
<soap:body use="literal" namespace="http://helloworldwebservice"/>
</output>
</operation>
<operation name="simpleHelloWorld">             Partie concrète
<soap:operation soapAction=""/>
<input>
<soap:body use="literal" namespace="http://helloworldwebservice"/>
</input>
<output>
<soap:body use="literal" namespace="http://helloworldwebservice"/>
</output>
</operation>
</binding>
<service name="HelloWorld">
<port name="HelloWorldPort" binding="tns:HelloWorldPortBinding">
<soap:address location="http://helloworldwebservice"/>
</port>
</service>
</definitions>
```

Figure 2.2 — Document WSDL du service *HelloWorld* [21].

OASIS et il est le résultat d'un accord interindustriel entre Microsoft, Ariba, IBM, Sun Microsystems, Oracle, HP, SAP, etc. Il offre un mécanisme de registre distribué de services permettant leur publication par les fournisseurs et leur découverte par les clients.

Les données stockés dans l'UDDI sont structurées en XML et organisées en trois parties connues sous le nom de pages :

- **Pages blanches** : fournissent des descriptions générales sur les fournisseurs de services à savoir le nom de l'entreprise qui fournit le service, son identificateur commercial, ses adresses, etc.

- **Pages jaunes** : comportent des descriptions détaillées sur les fournisseurs de services catalogués dans les pages blanches de façon à classer les entreprises et les services par secteurs d'activités.

- **Pages vertes** : procurent des informations techniques sur les services Web catalogués. Ces informations incluent la description du service, du processus de son utilisation et des protocoles utilisés pour son invocation.

Le protocole d'utilisation de l'UDDI offre trois fonctions de base :

- **publish** : permet de publier un nouveau service.

- **find** : permet d'interroger l'annuaire UDDI.

- **bind** : permet d'établir une connexion entre l'application cliente et le service.

Tout comme les services Web, il est possible de créer des annuaires UDDI privés (utilisés à l'intérieur de l'entreprise) ou publics (ouverts à l'ensemble de l'internet).

2.1.4 Architecture de référence

L'objectif de la mise en place d'une architecture pour la technologie service Web est de proposer un schéma directif et une vision sur le fonctionnement et la dynamique de cette nouvelle technologie. Pour cela, une architecture de référence a été mise en place par le W3C. Cette architecture a été proposée afin de promouvoir l'interopérabilité et l'extensibilité des services Web et de préserver ces deux objectifs lors des évolutions technologiques successives. Cette architecture est connue sous le nom AOS (Architecture Orientée Services) [31] [95]. Elle propose une perspective globale pour le développement, la gestion et le fonctionnement des services Web et elle s'articule autour des trois rôles suivants :

- **Le fournisseur** : correspond au propriétaire du service. Il fournit une plateforme d'accueil du service. Il a pour rôle de créer un service, de l'héberger et de le publier pour le mettre à la disposition des clients ou des partenaires.

– **Le client** : correspond au demandeur du service. Il est constitué par l'application qui va rechercher et invoquer un service. L'application cliente peut être elle même un service Web.

– **L'annuaire** : correspond à un registre de descriptions de services. Il offre des facilités de publication de services à l'intention des fournisseurs ainsi que des facilités de recherche de services à l'intention des clients.

Les interactions de base entre ces trois rôles incluent les opérations de transport, de description et de découverte et de transport qui sont assurées respectivement par les standards SOAP, WSDL et UDDI.

Nous décrivons, par la Figure 2.3, un scénario type d'utilisation de cette architecture.

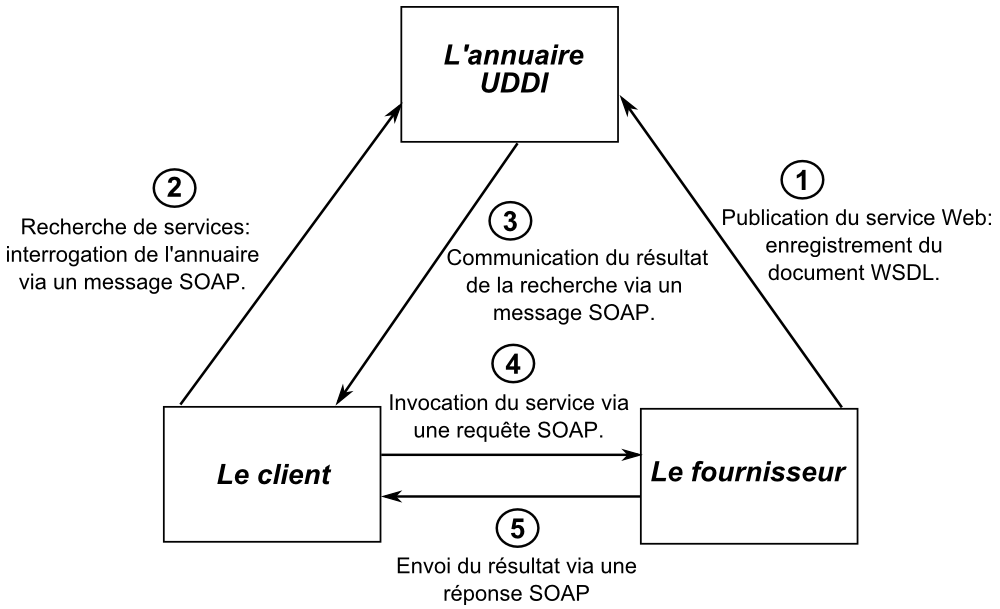


Figure 2.3 — Architecture de référence des services Web AOS.

Ce scénario se déroule en plusieurs étapes qui sont les suivantes :

1. Le fournisseur définit la description de son service dans un document WSDL et la publie dans l'annuaire UDDI.

2. Le client, désirant trouver un service, interroge l'annuaire UDDI via un message SOAP.

3. L'annuaire retourne, via un message SOAP aussi, une liste de services qui répondent à la requête du client. Le client n'a qu'à choisir un parmi la liste.

4. Le client récupère le document WSDL du service choisi. Ensuite, il examine ce document afin de récupérer les informations nécessaires lui permettant de se connecter au fournisseur et d'interagir avec le service considéré. Enfin, il invoque l'opération désirée par le biais d'une requête SOAP renfermant les paramètres d'entrée de l'opération.

5. Le service, du côté du fournisseur, reçoit la requête, la traite, formule la réponse SOAP et l'envoie au client.

2.2 La composition des services Web : Concepts, terminologie et technologies associées

Les services Web, tels qu'ils sont conçus, peuvent également se voir employés dans le cadre d'applications distribuées. En effet, en exploitant les standards ouverts du Web et en assurant un couplage faible des composants, la technologie service Web présente une approche flexible et universelle pour l'interopérabilité de systèmes hétérogènes. Ceci a pour effet de permettre une intégration des applications plus rapide, moins coûteuse et avec des perspectives prometteuses d'évolution et de réutilisation pour les entreprises. Cette opération d'intégration est appelée composition.

2.2.1 Définition

Selon Gardarin [37] : *"La composition est une technique permettant d'assembler des services Web afin d'atteindre un objectif particulier, par l'intermédiaire de primitives de contrôle (boucle, test, traitement d'exception, etc.) et d'échange (envoi et réception de messages). Les services composants existent au préalable et peuvent ne pas être fournis par la même organisation".*

Selon S. Dustdar et W. Schreiner [29] : *"L'infrastructure de base des services Web suffit pour la mise en œuvre d'interactions simples entre un client et un service Web. Si la mise en œuvre d'une application métier implique l'invocation d'autres services web, il est nécessaire donc de combiner les fonctionnalités de plusieurs services web. Dans ce cas, nous parlons d'une composition de services Web".*

La composition ou l'agrégation de services Web est une opération qui consiste à construire de nouvelles applications ou services appelés services composites ou agrégats par assemblage de services déjà existants nommés services basiques ou élémentaires.

La composition spécifie quels services doivent être invoqués, dans quel ordre et sous quelles pré-conditions. Les services basiques peuvent être soient des services atomiques soient des services composites.

La composition de services Web vise essentiellement quatre objectifs :

- Créer de nouvelles fonctionnalités en combinant des services déjà existants.
- Résoudre des problèmes complexes auxquels aucune solution n'a été trouvée.
- Faire collaborer plusieurs entreprises ensemble.
- Optimiser et améliorer une fonctionnalité existante.

2.2.2 Architecture étendue des services Web

La composition de services a donné naissance à une extension de l'AOS : l'architecture étendue [67] [69]. Cette architecture est constituée de plusieurs couches se superposant les unes sur les autres. De ce fait, elle est également appelée pile des services Web. Cette pile ne trouve pas encore de consensus quand à sa définition standardisée. La Figure 2.4 est une proposition d'une telle pile.

Les couches de cette architecture sont les suivantes :
- Couche d'infrastructure de base : renferme les fondements théoriques et technologiques établis par l'architecture de référence.
- Couche de processus métier : s'appuie sur la couche d'infrastructure de base et elle se préoccupe des aspects relatifs à la composition de services à savoir : la coordination, la supervision, la sémantique, etc.
- Couche de gestion : est la couche supérieure et elle s'intéresse à la configuration, au déploiement et à la maintenance des services Web, etc.

L'architecture étendue attache à chacune des ces couches des aspects spécifiques à l'ingénierie des services. Celle ci est définie comme étant l'activité qui consiste à analyser, modéliser et développer les techniques et les méthodologies nécessaires pour les approches à services basiques et/ou composites.

Outre les nouvelles couches de composition et de gestion, l'architecture étendue prend également en compte les aspects importants de propriétés comportementales et non-fonctionnelles d'un service qui ne sont pas explicitement abordée par l'architecture de référence. Pour cette fin, la couche d'infrastructure de base affine les exigences

décrites au niveau de l'architecture de référence pour y inclure également une description du comportement du service et de ses propriétés non-fonctionnelles.

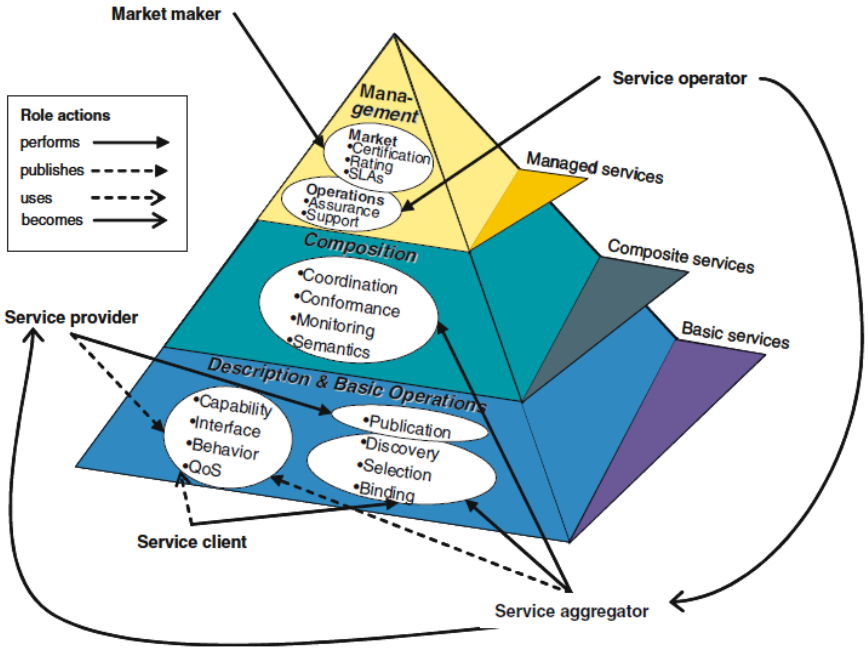


Figure 2.4 — Architecture étendue des services Web [69].

2.2.3 Composition statique vs composition dynamique

La composition des services Web peut être soit une composition statique soit une composition dynamique [29] :

- **La composition statique** : est appelée aussi composition off-line, précompilée ou encore proactive. C'est une composition qui utilise des services basiques qui sont au préalablement définis d'une façon figée et qui ne peuvent pas changer en fonction du contexte du client. Ce type de composition engendre des applications peu flexibles, parfois inappropriées avec les exigences des clients.

- **La composition dynamique** : appelée aussi composition on-line, postcompilée ou encore réactive. Elle se réfère à la sélection des services basiques "à la volée". Autrement dit, la sélection des services basiques ne peut pas être prédéfinie à l'avance mais elle sera faite au moment de l'exécution en fonction des contraintes imposées par le client. Ceci permet d'élaborer différents scénarii de composition qui offrent les mêmes fonctionnalités et qui tiennent compte de la dynamique de la situation du client.

Ces deux catégories de composition renferment deux sous-types :

- **La composition obligatoire** : est la situation où tous les services basiques doivent obligatoirement participer pour l'exécution de l'application. Ces services sont dépendants du succès de l'exécution d'autres services pour produire le résultat requis.

- **La composition optionnelle** : est une composition qui ne nécessite pas obligatoirement la participation de certains services basiques pour l'exécution de l'application.

Microsoft Biztalk et Bea WebLogic sont deux exemples de moteurs de composition statiques de services Web. Pour la composition dynamique, nous trouvons les plate-formes e-flow de HP et Sword de Stanford [90].

Dans le cadre d'une composition statique de services, si les fournisseurs proposent d'autres services ou changent les anciens services, des incohérences peuvent être causées. Ceci qui demande un changement de l'architecture du logiciel, voire de la définition de l'application et crée l'obligation de faire une nouvelle conception de l'application. C'est pourquoi, la composition statique des services web est considérée rigide et trop restrictive [86].

2.2.4 Procédés de coordination : Orchestration vs Chorégraphie

Le développement d'applications à base de services Web s'appuie sur la composition de ces derniers. Mais, le bon fonctionnement de ces applications met en évidence des problèmes d'interopérabilité et de coordination des services. Pour faire face à ces problèmes, différents canevas appelés procédés ont été proposés et mis en œuvre pour gérer et écrire la coordination des opérations des applications à base de services. Nous distinguons deux catégories de procédés de coordination : des procédés implémentant l'orchestration de services et des procédés implémentant la chorégraphie des services.

2.2.4.1 Orchestration

Selon Sanlaville [86] : *"L'orchestration des services Web permet de définir l'arrangement et l'enchaînement de ces services selon un canevas bien défini. Elle décrit la manière par laquelle les services peuvent interagir ensemble tout en incluant l'ordre d'exécution des différentes interactions".*

L'orchestration se base sur un procédé métier exécutable permettant de décrire l'enchaînement et les interactions des différents services basiques collaborant dans une composition. L'orchestration offre une vision centralisée ; le procédé est toujours contrôlé par l'un des partenaires métiers. Ce dernier joue le rôle d'un chef d'orchestre qui se

charge d'appeler les services de la composition suivant l'ordre d'exécution déjà défini par le processus métier. Le principe de l'orchestration est illustré par la Figure 2.5.

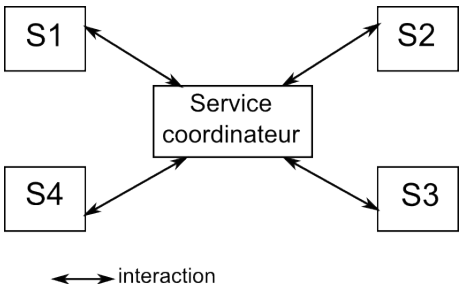


Figure 2.5 — Principe de l'orchestration des services Web.

2.2.4.2 Chorégraphie

Selon Sanlaville [86] : *"La chorégraphie permet de tracer la séquence de messages échangés dans un contexte de composition de services Web. Elle est typiquement liée à la description de conversations existantes entre les services tout en impliquant plusieurs parties, incluant les clients, les fournisseurs et les partenaires ".*

La chorégraphie est de nature plus collaborative que l'orchestration ; chaque participant impliqué dans le procédé décrit le rôle qu'il joue dans l'interaction. Ceci est explicité à travers la description des conversations échangées entre paires de services Web. Contrairement à l'orchestration, il n'y a pas de service coordinateur ; chaque service est conscient des services qui peuvent l'appeler ainsi que ceux qu'il doit appeler pour exécuter le processus métier. La chorégraphie offre donc une vision décentralisée. Le principe de la chorégraphie est illustré par la Figure 2.6.

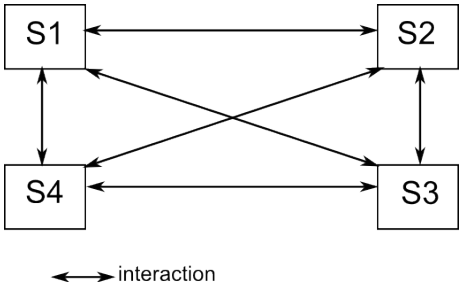


Figure 2.6 — Principe de la chorégraphie des services Web.

2.2.4.3 Avantages de l'orchestration

L'orchestration est considérée comme la meilleure approche de coordination de services Web puisqu'elle offre plusieurs avantages compétitifs, parmi lesquels nous citons [38] :

- **Avantage organisationnel** : réduction de l'écart entre la conception et l'implémentation ;
- **Avantage de gestion** : réduction des coûts de mise en œuvre en proposant l'utilisation de normes et de standards ouverts ;
- **Avantage stratégique** : Meilleure flexibilité ; les services Web peuvent être intégrés sans soucis parce qu'ils n'ont pas "conscience" d'appartenir à une composition ;
- **Avantages techniques** : Portabilité, simplicité et réutilisabilité du code.

2.2.4.4 Langage d'orchestration BPEL

Il existe plusieurs langages d'orchestration de services Web comme : BPEL [64], WSCI [94], XPDL [16], BPML [65], YAWL [92] et Orc [52], etc. Mais le langage le plus utilisé parmis ces derniers est BPEL. BPEL (Business Process Execution Language) est le langage d'orchestration le plus utilisé qui a remporté "la bataille des standards". Il est une spécification d'IBM, Microsoft, et BEA permettant la coordination des interactions entre l'instance du service composite et ses partenaires sous forme d'un schéma XML [64]. BPEL est appelé aussi WS-BPEL (Web Service Business Process Execution Language) ou BPEL4WS (Business Process Execution Language for Web Services). Ce langage est la fusion des spécifications XLANG [8] (XML Business Process Language) et WSFL [9] (Web Services Flow Language). BPEL contient les caractéristiques d'un langage structuré en blocs de XLANG, ainsi que les caractéristiques d'un graphe direct de WSFL.

BPEL4WS repose sur un modèle constitué d'activités de coordination qui peuvent être de deux types : de base ou structurées. Les activités de base permettent :

- d'invoquer une opération d'un service Web (**invoke**) ;
- de recevoir une requête (**receive**) ;
- de générer une réponse à une requête (**reply**) ;

8. Langage d'orchestration de services Web proposé par IBM.
9. Langage d'orchestration de services Web proposé par Microsoft.

– de copier une donnée (**assign**) ;

– de marquer un temps d'attente (**wait**) ;

– de terminer une activité (**terminate**).

Chaque activité de base est réalisée par une opération d'un service et chaque service est associé à un PartnerLink. Celui-ci contient les informations sur le rôle et l'interface PortType du service qui réalise cette opération. Les activités structurées utilisent les activités de base pour décrire :

– des séquences ordonnées (**sequence**) ;

– des exécutions en parallèle (**flow**) ;

– des branchements conditionnels (**switch, if**) ;

– des boucles (**while**) ;

– des chemins alternatifs (**pick**) ;

– des regroupements d'activités (**scope**).

Le langage BPEL intègre également un mécanisme de gestion des exceptions (**throw**), ainsi qu'un mécanisme de compensation (**compensationHandlers**).

Le langage BPEL est exécutable ; il est interprété par un moteur d'orchestration géré par l'un des participants de l'orchestration. Exemples de tels moteurs : TIBCO Business Works [10], Apache ODE [11], Oracle BPEL Process Manager [12], IBM WebSphere Process Server [13], etc.

2.2.4.5 Scénario d'orchestration : l'application "Agence de voyages"

Considérons une application à base de services "*Agence de voyages*". Cette application est formée par quatre services Web : "*S1 : Consultation vols*", "*S2 : Réservation vols*", "*S3 : Consultation hôtels*" et "*S4 : Réservation hôtels*". *S1* et *S2* sont offerts par une compagnie aérienne les deux autres services appartiennent à une chaîne hôtelière. Cette application permet d'organiser des formules de voyages selon une liste de préférences : vols et hôtels.

10. http ://www.tibco.com/products/soa/composite-applications/activematrix-businessworks/
11. http ://ode.apache.org/
12. http ://www.oracle.com/technetwork/middleware/bpel/overview/index.html
13. http ://www-01.ibm.com/software/integration/wps/

Le scénario d'orchestration de cette application est illustré par la Figure 2.7.

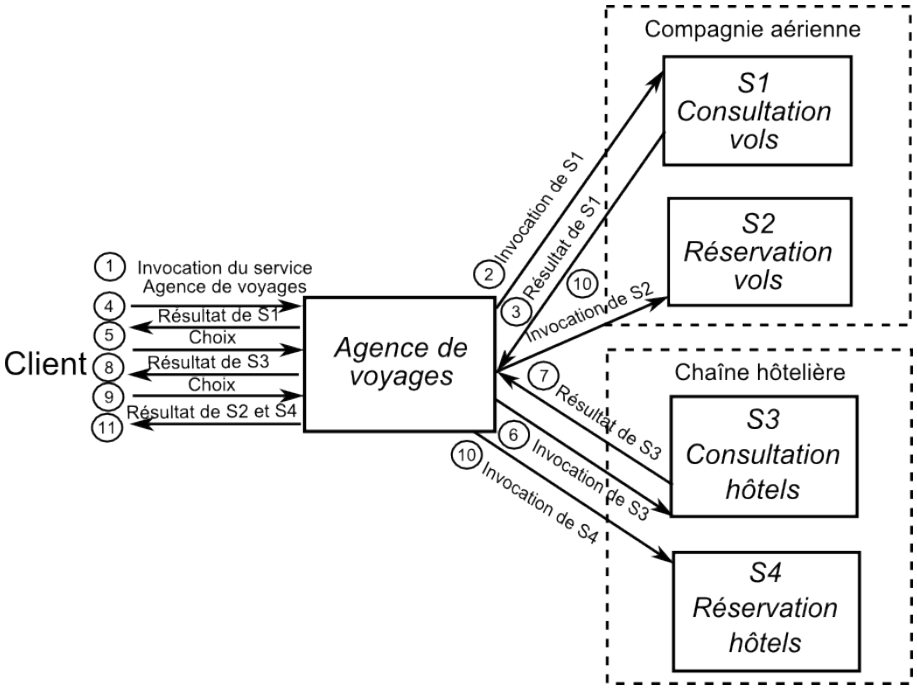


Figure 2.7 — Scénario d'orchestration : l'application "*Agence de voyages*".

Il se déroule comme suit :

1. Le service "*Agence de voyages*" reçoit les informations nécessaires à l'organisation du voyage (destination, date et heure de départ, date et heure de retour, etc.).

2. Le service "*Agence de voyages*" formule alors une requête et l'envoie au service S1 de consultations de vols.

3. Le service "*Agence de voyages*" reçoit la réponse de *S1*.

4. Le service "*Agence de voyages*" renvoie la liste des vols au client pour effectuer son choix.

5. Le client retourne son choix au service "*Agence de voyages*".

6. Le service "*Agence de voyages*" invoque le service *S2* de consultation d'hôtels pour vérifier les disponibilités.

7. Le service "*Agence de voyages*" reçoit la réponse de *S2*.

8. Le service "*Agence de voyages*" transmet la réponse au client.

9. Le client renvoie son choix au service "*Agence de voyages*".

10. Le service "*Agence de voyages*" effectue la réservation du vol et de l'hôtel.

11. Le service "*Agence de voyages*" envoie une confirmation de réservation au client.

La Figure 2.8 donne un extrait du code BPEL correspondant. Ce scénario d'orches-

```
...
<bpel:sequence name="Agence de voyage">
    <bpel:sequence name="Sequence">
        <bpel:invoke name="S1" inputVariable="input S1" outputVariable="output S1" partnerLink="PartnerLink S1"></bpel:invoke>
        <bpel:invoke name="S3" partnerLink="PartnerLink S3"></bpel:invoke>
    </bpel:sequence>
    <bpel:flow name="Flow">
        <bpel:invoke name="S2" partnerLink="PartnerLink S2" inputVariable="input S2" outputVariable="output S2"></bpel:invoke>
        <bpel:invoke name="S4" partnerLink="PartnerLink S4" inputVariable="input S4" outputVariable="output S4"></bpel:invoke>
    </bpel:flow>
</bpel:sequence>
...
```

Figure 2.8 — Extrait du code BPEL correspondant à l'application "*Agence de voyages*".

tration illustre la simplicité du procédé d'orchestration devant les apports qu'il procure en termes de coordination et d'interopérabilité des services.

2.2.5 Cycle de vie des applications à base de services

Le cycle de vie permet d'offrir un ensemble de directives et principes méthodologiques pour spécifier, construire, exécuter et contrôler les applications à base de services Web. La Figure 2.9 est une proposition d'un tel cycle [100] [68]. Ce cycle de vie est vu comme un processus itératif et incrémental qui comporte six phases :

- **Phase de planification** : aide à déterminer la série d'opérations de services qui doivent être trouvés et composées afin de satisfaire la requête des utilisateurs. Des modèles de domaine sont utilisés pour décrire les services en termes de besoins fonctionnels et non-fonctionnels des utilisateurs.

- **Phase de définition** : permet de définir le service composite d'une manière abstraite. La définition du service composite emploie WSDL en conjonction avec BPEL.

- **Phase de construction** : a pour rôle de donner des constructions concrètes pour les définitions abstraites fournies par la phase de définition, et ceci en

évaluant la composabilité des services, leurs capacités de conformité et la synchronisation et la priorité de leur exécution. Deux choix sont possibles pour la construction concrète du service composite : réutilisation de services déjà existants et/ou développement de nouveaux services.

– **Phase d'exécution** : permet d'implémenter, de déployer et d'exécuter le service composite.

– **Phase d'observation** : assure le suivi de l'exécution des services.

Une fois cette boucle faite, la composition construite doit pouvoir évoluer afin de satisfaire les besoins des utilisateurs. Cette évolution peut se produire à chaque étape du cycle de vie : une fonctionnalité peut être rajoutée, un service peut être supprimée, un fournisseur peut se joindre à la coopération, des liens entre les services peuvent être modifiés, etc.

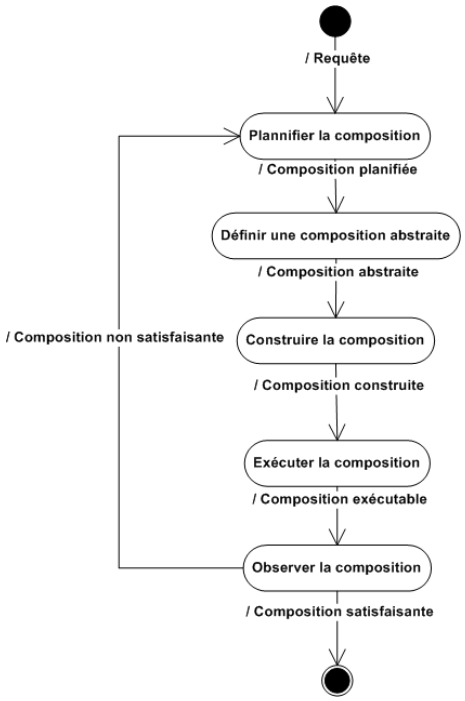


Figure 2.9 — Cycle de vie des applications à base de services Web.

2.3 Synthèse et problématique relevée

Pour être en phase avec le monde de l'entreprise, les applications à base de services doivent être modélisée, conçues et développées de façon à pouvoir s'adapter à différents

types d'utilisateurs ayant des exigences différentes. Cette constatation est tirée suite au changement de comportement des utilisateurs qui ne veulent plus s'adapter aux services qu'ils utilisent mais qui préfèrent que ces derniers soient configurés selon leurs besoins [8]. Cette situation a fait émerger la notion de variabilité. La variabilité est traduite, dans le monde des services Web, grâce au procédé dynamique de composition qui permet l'élaboration de différents scénarii s'approriant chacun avec des exigences particulières d'utilisateurs. Ainsi, augmenter l'efficacité des applications à base de services revient à proposer des services adaptables : couvrant le plus grand nombre de besoins utilisateurs [8]. Pour atteindre cette fin, il est nécessaire d'opter pour un changement d'échelle et adopter une vision "centrée exigences" de la composition où les services sont considérés comme des interactions "métiers" décrits en termes d'exigences et non plus en termes de fonctionnalités. Ceci garanti un "haut niveau" d'abstraction qui renforce la variabilité et la réutilisation des services. La problématique qui se pose à ce niveau est : **Comment assurer l'alignement entre les services métiers et les services logiciels qui les réalisent tout en préservant la variabilité de composition ?**

Plusieurs questions découlent de cette problématique :

- Quel est le modèle d'exigences le plus approprié à la description des services métiers et à l'expression de leur variabilité ?
- Comment préserver la variabilité de composition tout au long des différentes phases du cycle de développement des applications à base de service ?
- Comment assurer l'alignement entre les services métiers et les services logiciels qui les réalisent ?
- Comment valider cet alignement afin de garantir la satisfaction des utilisateurs ?

Pour ce faire, nous sommes donc ramenés à dévoiler la notion de variabilité afin d'opter pour le modèle d'exigences décrivant le mieux cette propriété. Nous nous intéresserons, également, à élucider la problématique d'alignement des services afin de décider de l'approche permettant sa réalisation.

2.4 Conclusion

Dans ce chapitre, nous avons présenté les concepts de base des services Web et de leur composition. Nous avons, également, détaillé la problématique de ce travail : **Comment assurer l'alignement entre les services métiers et les services logiciels qui les réalisent tout en préservant la variabilité de composition ?**

Dans le chapitre qui suit, nous présentons l'état de l'art portant sur les approches "centrées exigences" de composition de services Web.

Chapitre 3

Approches centrées exigences de composition de services Web

Ce chapitre commence par une introduction des deux concepts : alignement et variabilité, ensuite, présente un état de l'art sur les approches centrées exigences de composition de services Web. La plupart des travaux existants se sont focalisés sur l'alignement ou la variabilité de façon isolée ; peu nombreux sont les travaux qui ont abordé ces deux aspects dans le cadre de développement d'applications par composition de services Web. À la fin du chapitre, comme bilan, nous comparons ces approches d'état de l'art afin d'identifier leurs limitations et mettre en évidence les contributions apportées par notre approche.

Sommaire

3.1 L'alignement : atouts, définition et cadre de référence

Selon G. Regev et A.Wegmann [76] "*l'alignement se définit comme la correspondance entre un ensemble de composants* ".

J-D. McKeen et H-A. Smith [58] précisent que "*l'alignement des technologie d'information existe lorsque les buts, les activités et les processus métier de l'entreprise sont en harmonie avec les systèmes d'information qui les supportent*".

Enfin, B-H. Reich et I. Benbasat [77] conçoivent l'alignement comme "*le degré selon lequel la mission, les objectifs, et les plans contenus dans la stratégie métier sont partagés et supportés par la stratégie des technologies d'information*".

À partir de ces trois définitions, nous déduisons que l'alignement est une démarche qui met les buts de l'entreprise en harmonie, en correspondance, avec les processus métier et les systèmes qui les supportent.

Les approches d'alignement sont classées en trois catégories : approche top-down, approche bottom-up et approche mixte [32] [91].

- **Approche top-down.** C'est une approche qui consiste à identifier et à analyser un ensemble de buts qui sont ensuite mis en œuvre. Cette approche repose sur les modèles de buts proposés en ingénierie d'exigences pour capter les stratégies métier de l'entreprise et sur des règles de mise en correspondance pour opérationnaliser les buts par des spécifications opérationnelles du système. Cette approche répond à la question "Comment".

- **Approche bottom-up.** C'est une approche qui consiste à identifier les buts définissant le métier de l'entreprise à partir des spécifications opérationnelles décrivant les systèmes. Cette approche s'intéresse au "Pourquoi".

- **Approche mixte.** C'est une approche qui combine les deux approches précédentes. Dans sa globalité, elle est une approche top-down mais l'alignement entre les deux niveaux stratégique et opérationnel est analysé et validé d'une manière bottom-up.

Le cadre de référence de l'alignement considère quatre vues : la vue "Objet", la vue "But", la vue "Méthode" et finalement la vue "Outil". Ce cadre de référence est représenté par la Figure 3.1.

La vue "Objet" s'intéresse aux entités que nous désirons aligner et aux liens qui réalisent leur alignement. Cette vue répond à la question "Quoi". La vue "But"

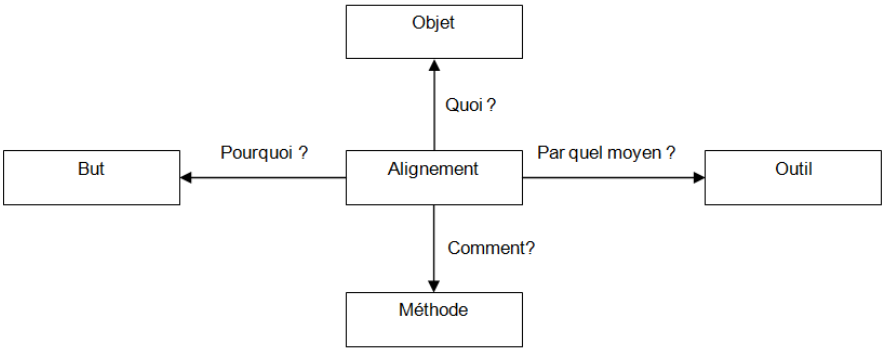


Figure 3.1 — Les quatre vues du cadre de référence de l'alignement [32].

identifie les buts de l'approche d'alignement et elle répond à la question "Pourquoi". La vue "Méthode" explicite la méthode de définition des spécifications opérationnelles qui répondent aux buts stratégiques. Elle s'intéresse à la question "comment". Finalement, la vue "Outil" décrit les outils utilisés pour la mise en œuvre de l'approched'alignement. Elle répond à la question "Par quel moyen".

L'objectif de notre thèse est d'assurer un alignement entre les applications à base de services Web développées et les exigences de l'entreprise. Mais également, elle vise à intégrer la variabilité tout au long des phases de développement de ces applications afin de proposer des solutions génériques qui couvrent le plus grand nombre d'exigences utilisateurs. Dans ce qui suit, nous donnons la définition de la variabilité et nous détaillons ses différents types.

3.2 La variabilité : définition et types

Selon J. Van Gurp *et al.* [40], la variabilité est définie dans le contexte des systèmes logiciels comme étant : "*la capacité d'un système ou d'un artefact à être étendu, changé, personnalisé ou configuré selon un contexte d'utilisation particulier* ".

La variabilité traduit l'aptitude d'un système à s'adapter, se spécialiser et se configurer en fonction du contexte. L'objectif essentiel de la variabilité est donc l'adaptation du système aux besoins des utilisateurs. Ainsi, un haut degré de variabilité implique une augmentation de la capacité du système à être utilisé dans des contextes d'utilisation très variés.

La variabilité est généralement exprimée en termes de points de variations et de variantes [39].

– Un point de variation désigne un endroit où il existe une variation. C'est à cet endroit que des choix doivent être fait afin d'identifier les variantes à utiliser.

– Chaque variante est une manière de réalisation de la variabilité.

Nous distinguons deux catégories de variabilité selon la classification proposée par [42] : la variabilité essentielle et la variabilité technique.

– La variabilité essentielle représente le point de vue de l'utilisateur qui s'intéresse aux aspects relatifs à l'utilisation du système tels que les exigences fonctionnelles et non-fonctionnelles auxquels le système répond.

– La variabilité technique correspond au point de vue du développeur qui s'intéresse aux aspects relatifs à la réalisation de la variabilité tels que les techniques et les outils d'implémentation utilisés pour la mise en œuvre du système.

La variabilité essentielle décrit ce qui doit être implémenté alors que la variabilité technique définit comment implémenter la variabilité [8]. Ces deux catégories se subdivisent en sous catégories qui sont décrites par la Figure 3.2 et la Figure 3.3.

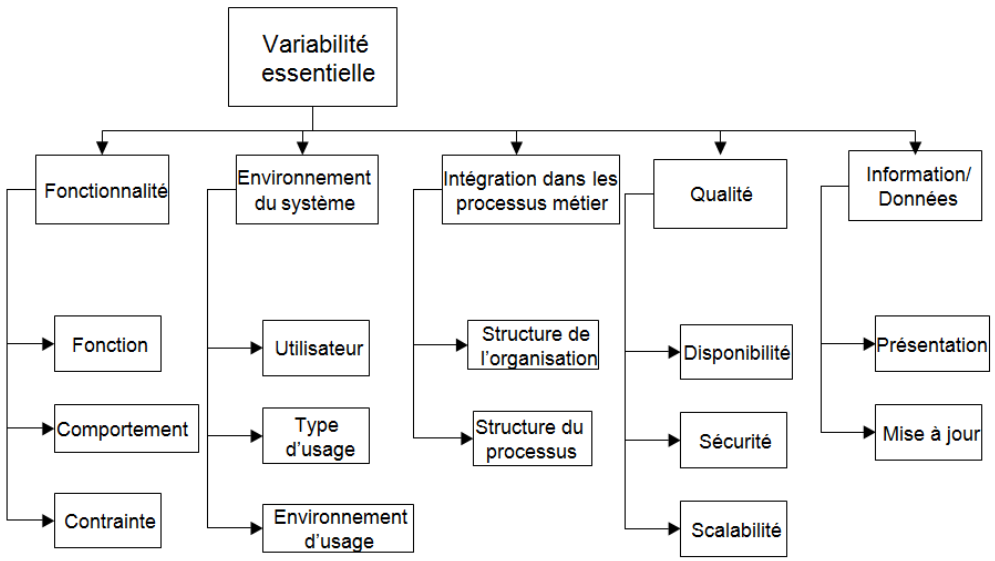


Figure 3.2 — Les catégories de la variabilité essentielle selon [42].

En se référant à littérature qui relate aux applications à base de services Web [89], nous avons pu distinguer deux facettes de variabilité essentielle qui sont les variabilités fonctionnelles et non-fonctionnelles. Ces deux facettes sont liées à la phase d'expression des exigences. Pour la variabilité technique, elle implique deux facettes égalment : la

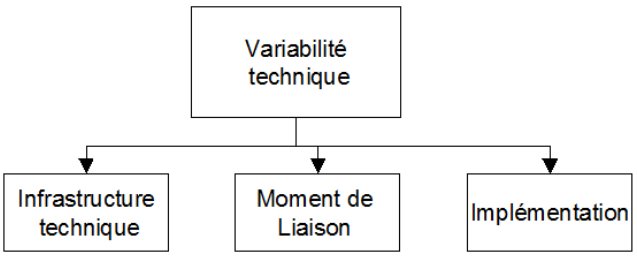


Figure 3.3 — Les catégories de la variabilité techniques selon [42].

variabilité de coordination et d'exécution. Ces deux facettes sont liées respectivement aux phases de coordination et d'exécution des services logiciels.

Dans les travaux existants, nous avons pu constater que les deux catégories de la variabilité, essentielle et techniques, ont été traitées d'une façon isolée et que la variabilité technique a suscité plus d'intérêt que la variabilité essentielle. L'objectif de notre thèse est de coupler la variabilité essentielle à la variabilité technique tout au long des phases de développement des applications à base de services tout en assurant un alignement entre ses applications et les exigences des utilisateurs.

Dans ce qui suit, nous exposons les différents travaux qui ont opté pour une vision centrée exigences pour la composition de services tout en se considérant une classification se basant sur la nature de l'approche utilisée pour l'expression des exigences.

3.3 Les approches centrées exigences de composition de services Web

Dans la littérature afférente à la composition centrée exigences des services Web, nous distinguons trois catégories d'approches d'expression des exigences : les approches formelles, les approches semi-formelles et les approches informelles. Dans ce qui suit, nous exposons les travaux liés à ces différentes approches.

3.3.1 Les approches formelles

Ceux sont des approches qui reposent sur des structures et des langages formels comme : les réseaux de Pétri, les ontologies, les algèbres de processus, les théories temporelles, etc.

Les réseaux de Pétri. Les réseaux de Pétri sont des modèles mathématiques servant à la modélisation de systèmes divers. Ils sont représentés par des graphes bipartis reliant des places (les nœuds) et des transitions (les liens). Dans le cadre de la composition centrée exigences des services Web, les réseaux Pétri ont fait l'objet de travaux comme [43] et [53].

Dans [43], R. Hamadi et B. Benatallah proposent de formaliser les compositions de services en utilisant les réseaux de pétri. Cette approche permet l'expression d'opérateurs liés à la gestion des flux de contrôle comme : la séquence, l'alternative, l'itération, le parallélisme, la discrimination ou encore la sélection et ceci en utilisant un ensemble de règles définies par une grammaire BNF. Le réseau de Pétri obtenu décrit une orchestration de services qui peut ensuite être analysée et même comparée avec d'autres réseaux de Pétri.

Les réseaux de Pétri font aussi l'objet de travaux de J-J. Le et F. He [53] visant à sélectionner et composer automatiquement les services Web en utilisant les réseaux de Pétri. L'approche proposée permet de traduire les services Web disponibles dans un annuaire local et décrits en OWL-S en un ensemble de règles de production qui sont transformés en un modèle de réseaux de Pétri. En se basant sur ce modèle et sur un ensemble de règles de dépendance entre services, un algorithme formel est proposé pour assurer d'une manière automatique la sélection et la composition de services qui répondent au mieux aux exigences des utilisateurs qui sont formulées en termes de paramètres d'entrée/sortie.

Les ontologies. Une ontologie est un modèle de données représentatif d'un ensemble de concepts liés sémantiquement à un domaine, ainsi que des relations entre ces concepts. Les ontologies ont été utilisées dans le cadre de la composition centrée exigences des services Web. Nous présentons dans ce qui suit les travaux [3], [44], [87], [99] et [93].

Arpinar *et al.* [3] proposent une approche semi-automatique permettant la découverte et la composition de services Web. L'approche proposée utilise des descriptions et un processus ontologique pour découvrir et composer les services dont la sémantique s'aligne avec les exigences des utilisateurs. Cette approche est mise en œuvre par un système qui utilise l'ontologie DAML-S et une ontologie de processus qui la complète pour décrire la requête de l'utilisateur. Dans cette requête, le service composite à développer est décrit en termes de paramètres d'entrée, paramètres de sortie et relations entre services qui le composent. Une fois la requête envoyée, elle est analysée et traitée en découvrant, composant, exécutant et supervisant les services qui répondent à cette requête. La sélection des services est faite en se référant à leur degré de composabilité qui équivaut au degré de similarité sémantique de leurs paramètres d'entrée et de sortie.

Dans [44], L. Hou *et al.* propose un cadre de travail permettant la découverte et la composition automatiques des services Web. Ce cadre de travail repose sur un modèle ontologique décrivant l'environnement d'un service Web en termes de ressources et d'interactions. Ce modèle étend l'ontologie OWL-S. Une fois les services sont découverts et composés, l'algèbre ?-calculus est utilisée pour vérifier la satisfaction des exigences.

E. Sirin *et al.*, dans [87], proposent le système SHOP2 qui permet d'assurer la composition automatique des services Web. Dans ce système, les exigences sont exprimées en utilisant l'ontologie OWL-S. Les descriptions ontologiques obtenues sont ensuite transformées en des problèmes de planification HTN (Hierarchical Task Network) à l'aide desquelles SHOP2 assure la composition, l'exécution et la supervision des services.

L'ontologie a été également utilisée dans [99] pour éliciter et modéliser les exigences des utilisateurs pour la composition de services Web. Dans le cadre de ce travail, Xiang *et al.* proposent le mécanisme SREM (Service Requirements Elicitation Mechanism) renfermant : (i) l'ontologie SRMO (Service Requirements Ontology) décrivant un service en termes de buts, d'acteurs, de tâches et de qualités, (ii) le processus SREP (Service Requirements Elicitation Process) permettant la génération de modèles d'exigences basée sur l'ontologie SRMO et (iii) le processus SRRP (Service Requirements Reconciliation Process) permettant de faire correspondre les services aux exigences élicités par le processus SREP.

Un travail récent utilisant les ontologies dans le cadre des applications à base de services est celui de Verlaine *et al.* [93]. Ce travail propose une approche permettant l'alignement des exigences des utilisateurs aux spécifications opérationnels de services WSDL et aux spécifications de qualité WSLA. Verlaine *et al.* définissent pour cela une ontologie permettant l'expression des exigences fonctionnels et non-fonctionnels. Pour réaliser cet alignement, un ensemble de règles de transformation définis en logique formelle *SIN* sont utilisés. L'outil STR@WS est proposé pou la mise en œuvre de cette approche.

Les théories temporelles. Ces formalismes ont été introduits pour représenter des connaissances temporelles ou pour spécifier des domaines d'objets dynamiques. Des travaux comme [56] et [83] ont exploité les théories temporelles pour assurer une description centrée exigences de la composition de services Web.

Dans [83], K. Mahbub et G. Spanoudakis proposent un cadre de travail permettant la supervision des exigences liées à des applications à base de services. Les exigences sont liées à des propriétés comportementales exprimées en une théorie temporelles qui est EC (Event Calculus). Les applications à base de services supervisées sont spécifiées

en langage de coordination BPEL. La supervision permet la détection de violations des propriétés décrites par les exigences.

[83] est une extension du travail de K. Mahbub et G. Spanoudakis [83]. L'approche proposée par Rouachid *et al.* permet, dans un premier temps, la formalisation d'une orchestration BPEL grâce à une étape de transformation qui rend possible la représentation d'une orchestration sous la forme d'un ensemble de prédicats décrits en EC (Event Calculus). Cette approche permet ensuite la vérification de propriétés comportementales à l'aide du prouveur automatique de théorèmes SPIKE avant et pendant l'exécution de l'orchestration. Ces propriétés étant elles-mêmes exprimées comme un ensemble de prédicats décrits en EC.

3.3.2 Les approches semi-formelles

Ce sont des approches qui ont emprunté des méthodologies et des formalismes du domaine de l'ingénierie des exigences pour identifier et analyser les exigences des utilisateurs décrivant les services à composer. Principalement, quatre formalismes ont été utilisés qui sont : Tropos, KAOS, i* et la Carte.

Tropos. Tropos [12] est une méthodologie de développement orientée agents qui couvre entièrement le cycle de développement logiciel : de l'ingénierie des exigences à l'implémentation. Cette méthodologie est en cinq phases : (i) ingénierie des premières exigences, (ii) ingénierie des exigences avancées, (iii) conception architecturale, (iv) conception détaillée, et (v) implémentation. Les principaux concepts clés de Tropos sont : le concept "Acteur" qui présente un agent (physique ou logiciel) et qui a comme rôle la description du comportement de cet agent dans un contexte précis et le concept "But" qui modélise l'intérêt d'un acteur et qui peut être de deux types différents : "But flou" n'ayant pas de définition exacte et/ou de critère permettant de savoir s'il a été satisfait ou non ou "But exact" qui est le contraire de "But flou".

Dans le cadre des applications à base de services, les travaux présentés par R. Kazhamiakin *et al.* [51] et Pistore *et al.* [72] proposent de dériver des compositions de services par affinement de modèles d'exigences Tropos.

L'approche proposée dans [72] permet d'enrichir les modèles formels Tropos avec du code BPEL et d'exploiter des techniques de vérification formelles offertes par l'outil SPIN sur les modèles Tropos pour assurer la vérification des compositions de services. A la différence de [72], Kazhamiakin *et al.* [51] utilisent le langage de spécification Promela pour décrire l'orchestration de services et non pas BPEL, mais la vérification est réalisée également en utilisant l'outil SPIN.

Le travail d'Aiello *et al.* [1] étend les travaux précédents en proposant d'assurer la

modélisation et la vérification non seulement des exigences fonctionnelles mais aussi des exigences non-fonctionnelles. La modélisation de ces exigences est réalisée moyennant la méthodologie Tropos. La vérification est basée sur deux types de raisonnements : quantitatif et qualitatif. Ces raisonnements permettent de vérifier la satisfaction de propriétés de qualité comme la sécurité exprimées en termes d'exigences non-fonctionnelles.

KAOS. KAOS [20] est une méthodologie orientée buts qui préconise une phase de modélisation des exigences avant d'écrire le cahier des charges. Cette méthodologie est menée de deux niveaux de formalismes complémentaires : un niveau semi-formel utilisé dans les premières phases d'analyse pour établir la structure générale du modèle d'exigences et un niveau formel permettant de raffiner le modèle semi-formel obtenu afin de le compléter, de le rendre consistant et de mieux le structurer. KAOS fédère 4 vues complémentaires et liées sur le système : (i) le modèle des buts qui spécifie les objectifs poursuivis par les différents agents impliqués, les exigences qui sont nécessaires pour atteindre ces objectifs et les obstacles qui empêchent la réalisation de ces objectifs, (ii) le modèle des responsabilités qui fournit un inventaire des agents humains ou automatisés qui interagissent avec le système et qui sont responsables de la satisfaction des exigences, (iii) le modèle objet qui décrit le comportement que les agents doivent entreprendre pour atteindre ou respecter les exigences dont ils sont responsables et finalement (iv) le modèle des opérations qui identifie le domaine couvert sous forme de concepts du domaine et de relations entre ces concepts.

Dans [79], N. Robinson propose un cadre de travail permettant d'éliciter les exigences, les raffiner et les superviser en utilisant la méthodologie KAOS. Les exigences sont exprimées tout d'abord en langage naturel puis modélisées en utilisant le modèle semi-formel KAOS. La supervision, assurée par ce cadre de travail, permet de détecter les éventuels conflits qui existent entre les services composés et les exigences élicités. Cette supervision est réalisée en utilisant le modèle formel de détection d'obstacles offert par KAOS.

G. Diaz *et al.* [22] utilisent la méthodologie KAOS pour modéliser des propriétés temporelles et appliquent la technique de Model Checking sur les chorégraphies de services Web pour la vérification de ces propriétés. Ils proposent une approche qui consiste en trois étapes : (i) dans la première étape, les exigences sont élicitées et modélisées en utilisant KAOS, (ii) dans une deuxième étape, ces exigences sont traduites en un langage de chorégraphie qui est WS-CDL, finalement (iii) les propriétés temporelles exprimées formellement par KAOS sont vérifiées par application de la technique du model checking sur les chorégraphies transformées en automates temporels.

La Carte. La Carte [82] est un formalisme qui s'inscrit dans le domaine d'ingénierie des méthodes qui est une discipline de conceptualisation, de construction et d'adaptation de méthodes, de techniques et d'outils pour le développement des systèmes

d'information. C'est une approche dite stratégique ; elle modélise les méthodes en termes d'intentions à réaliser afin de construire le système et de stratégies à suivre pour réaliser ces intentions. Puisqu'elle se base sur le concept "intention" qui exprime un but à atteindre, la Carte fait partie aussi des formalismes de l'ingénierie des exigences.

Le formalisme la Carte a été utilisé par Rolland *et al.*. [80] pour décrire les applications à base de services. La contribution principale de ce travail est la proposition du modèle intentionnel de services MIS qui permet de spécifier les services et leur composition d'une manière intentionnelle. Un ensemble de directives est défini pour l'extraction de ces services intentionnels et leur composition à partir des cartes qui les modélisent. Dans [81], Rolland *et al.* présentent une continuité de [80]. Dans le cadre de ce travail, une extension du modèle MIS est proposée afin de décrire des aspects de QdS liés aux services intentionnels. Aussi, propose-t-il une méthodologie basée sur un modèle opérationnel de services et un ensemble de règles de transformation pour l'opérationnalisation des services intentionnels.

Mirbel *et al.* [60] utilisent aussi le formalisme la Carte pour la modélisation des exigences des utilisateurs. Ce travail propose une approche sémantique guidée par les intentions pour la modélisation et la recherche de services. Mirbel *et al.* exploitent les capacités de raisonnement des modèles et langages du web sémantique pour dériver les caractéristiques des services web recherchés qui correspondent aux besoins des utilisateurs.

i*. i* [101] est une approche d'ingénierie d'exigences orientée agents. Les agents en i* sont associés à des buts qui peuvent être des buts individuels ou partagés. L'objectifs de i* est de représenter les aspects intentionnels et les aspects de configurations organisationnelles liés au système d'information. Il est formé par deux modèles : un modèle de raisonnement stratégique représentant les aspects intentionnels (raisonnements stratégiques) et un modèle de dépendances stratégiques représentant les acteurs qui interviennent et leurs dépendances.

Dans le cadre des applications à base de services, Pérez *et al.* [75] utilisent le formalisme i* pour la modélisation des exigences des utilisateurs et proposent un mécanisme semi-automatique permettant la découverte des services web qui réalisent ces exigences. L'approche proposée est réalisée sur quatre étapes. La première étape consiste à éliciter et analyser les exigences des utilisateurs formulées sous forme de requêtes. Ensuite, Pérez *et al.* proposent de modéliser et spécifier ces exigences en utilisant le formalisme i*. La deuxième étape exploite les ontologies (une ontologie de domaine et une ontologie d'application) pour la normalisation des exigences spécifiés en i*. La dernière étape permet la découverte des services logiciels. Ceci est fait par une recherche par mots clés et par catégories. L'approche est appliquée sur un registre privé de services de bioinformatique.

3.3.3 Les approches informelles

Ce sont des approches qui utilisent une formalisation en langage naturel pour l'expression des exigences des utilisateurs. Dans ce qui suit nous détaillons deux travaux [102] et [19] qui ont opté pour ce type d'approche pour la composition centrée exigences de services Web.

Le travail de Zachos *et al.* [102] a comme objectif d'aligner les exigences des utilisateurs aux services opérationnels disponibles. L'idée-clé de ce travail est de créer un cahier des charges spécifiant les exigences des utilisateurs, transformer ces exigences en des requêtes pour l'interrogation de l'annuaire de services et modifier le cahier des charges élaboré initialement en fonction des services retournés. La spécification des exigences est faite en langage naturel en utilisant le modèle de spécification d'exigences VOLERE. Les spécifications obtenues sont analysées par l'outil lexical WordNet pour enlever les ambiguïtés sémantiques. Pour découvrir les services logiciels qui réalisent les exigences des utilisateurs, ces spécifications sont transformées en des requêtes en utilisant le langage XQuery. Ces requêtes sont utilisées pour extraire les spécifications WSDL qui sont sémantiquement similaires aux exigences formulées à partir de l'annuaire de services. A la fin, Zachos *et al.* proposent de réaligner les exigences initialement formulées aux services Web trouvés.

Dans [19], Cremene *et al.* proposent également une approche informelle pour la composition de services Web. Cette approche se base sur une formulation en langage naturel des exigences des utilisateurs. Une fois saisies, ces exigences sont traitées par des outils de traitement automatique de langage naturel afin de les réécrire formellement sous la forme de requêtes. Ces requêtes sont utilisées pour sélectionner et composer les services logiciels. Comme dans [102], la sélection est faite en mesurant la similarité sémantique entre les requêtes qui expriment les exigences des utilisateurs et les spécifications WSDL des services. Pour la composition, des patrons sont définis pour l'assemblage des services sélectionnés. Cette approche est mise en œuvre par l'outil proposé NLSC (Natural Language Service Composer).

3.4 Bilan

Les travaux cités dans cet état de l'art présentent des approches assurant la composition centrée exigences de services Web. Cependant, chacune de ces approches se focalise sur un sous-ensemble des étapes nécessaires pour construire de manière systématique une application à base de services Web. En effet, et comme le montre le Tableau récapitulatif 3.1, aucune des approches présentées n'accompli toutes les phases liées au cycle de vie de développement des applications à base de services.

Concernant l'alignement, seulement quelques travaux comme [3], [44], [80], [81], [60]

et [19] se sont focalisés sur ce problème et ont proposé des approches qui ont opté pour l'alignement top-down en faisant correspondre les services logiciels aux exigences. Uniquement le travail de M. Zachos *et al.* [102] s'est intéressé à l'alignement bottom-up en proposant le raffinement des exigences initialement formulées en fonction des services logiciels découverts.

Peu nombreux sont aussi les travaux qui ont intégré la variabilité tout au long des étapes de construction des applications à base de services. Nous trouvons uniquement les travaux de C. Rolland *et al.* [80] et [81] qui ont tenu compte de trois facettes de la variabilité qui sont : la variabilité fonctionnelles, la variabilité de coordination et la variabilité d'exécution. Le travail de M. Cremene *et al.* [19] s'est intéressé uniquement de deux facettes de la variabilité qui sont : la variabilité fonctionnelle et la variabilité de coordination. Le reste des travaux comme [43], [53], [3], [87] et [99] se sont focalisés seulement sur la variabilité de coordination, d'autres comme [1], [60] et [75] sur la variabilité fonctionnelle.

Pour l'expression des exigences, la plupart des travaux de l'état de l'art ont considéré les exigences fonctionnelles et ont négligé les exigences non-fonctionnelles. Seuls les travaux [93], [1], [79] et [81] ont intégré des contraintes de QdS dans leurs modèles d'expression d'exigences.

Bien que la découverte et la sélection des services logiciels soient des étapes nécessaires pour assurer l'alignement entre les exigences et les services logiciels qui les réalisent, nous trouvons que quelques travaux qui proposent la mise en œuvre de ces deux étapes [53], [3], [44], [75], [102] et [19].

Pour la coordination des services logiciels, elle est toujours faite en boîte noire [56], [83], [72] et [22]. En effet, tous ces travaux n'explicitent pas le passage du modèle des exigences au procédé de coordination des services logiciels.

Finalement, la validation est faite toujours d'une manière partielle. Seuls [44] et [75] proposent la validation des exigences fonctionnelles, [1], [79] et [22] valident les exigences non-fonctionnelles.

3.5 Conclusion

Cette étude comparative et les différentes limitations soulevées nous ont amené à proposer une nouvelle approche de composition de services Web qui assure l'alignement entre les exigences fonctionnelles et non-fonctionnelles des utilisateurs et les services logiciels qui les réalisent tout en considérant les différentes facettes de la

variabilité à savoir : la variabilité fonctionnelle, non-fonctionnelle, de coordination et d'exécution. De plus, cette approche intègre les étapes de découverte, de sélection et de coordination des services logiciels et assure une validation permettant la vérification de la satisfaction des exigences fonctionnelles et non-fonctionnelles des utilisateurs.

Dans le chapitre 4, nous détaillons notre approche tout en essayant de fournir des éléments de réponse aux limitations soulevées au niveau des travaux de l'état de l'art.

			Alignement	Variabilité	Exigences fontionnelles/ non-fonctionnelles	Découverte des services logiciels	Sélection des services logiciels	Coordination des services logiciels	Validation
Approches formelles	**Réseaux de Pétri**	*R. Hamadi et B. Benatallah (2003)* **[43]**	-	Variabilité de coordination	-	-	-	-	Algèbre de réseaux de Pétri (propriétés comportementales)
		J-J. Le et F. He (2008) **[53]**	-	Variabilité de coordination	-	-	Algorithme de sélection	-	-
	Ontologies	*I-B. Arpinar et al. (2004)* **[3]**	Alignement top-down	Variabilité de coordination	Exigences fonctionnelles	Découverte par formulation de requêtes écrites en DAML-S	Similarité sémantique des paramètres d'entrée/sortie	-	-
		L. Hou et al. (2004) **[44]**	Alignement top-down	-	Exigences fonctionnelles	Algorithme de découverte	Algorithme de sélection	-	Algèbre Π-claculus (exigences fonctionnelles)
		E. Sirin et al. (2004) **[87]**	-	Variabilité de coordination	Exigences fonctionnelles	-	-	-	-
		J. Xiang et al. (2007) **[99]**	-	Variabilité de coordination	Exigences fonctionnelles	-	-	-	-
		B. Verlaine et al. (2010) **[93]**	Alignement top-down	-	Exigences fonctionnelles et non-fonctionnelles	-	-	-	-
	Théories temporelles	*K. Mahbub et G. Spanoudakis (2005)* **[56]**	-	-	-	-	-	BPEL (Génération en boîte noire)	Monitoring (propriétés comportementales)
		M. Rouached et C. Godart [2007] **[83]**	-	-	-	-	-	BPEL (Génération en boîte noire)	Analyse formelle par un prouveur automatique de théorèmes (propriétés comportementales)
Approches semi-formelles	**Tropos**	*R. Kazhamiakin et al. (2004)* **[51]**	-	-	Exigences fonctionnelles	-	-	-	Analyse formelle par l'outil SPIN (propriétés temporelles)
		M. Pistore et al. (2004) **[72]**	-	-	Exigences fonctionnelles	-	-	BPEL (Génération en boîte noire)	Analyse formelle par l'outil SPIN (propriétés temporelles)
		M. Aiello et P. Giorgini(2004) **[1]**	-	Variabilité fonctionnelle	Exigences fonctionnelles et non-fonctionnelles	-	-	-	Raisonnements quantitatif et qualitatif (exigences non-fonctionnelles)

	KAOS	*N. Robinson (2005)* **[79]**	-	-	Exigences fonctionnelles et non-fonctionnelles	-	-	-	Monitoring (exigences non-fonctionnelles)
		G. Diaz et al. (2006) **[22]**	-	-	-	-	-	WS-CDL (Génération en boîte noire)	Model checking (exigences temporelles)
	La Carte	*Rolland et al. (2007)* **[80]**	Alignement top-down	Variabilité fonctionnelle + Variabilité de coordination + Variabilité d'éxécution	Exigences fonctionnelles	-	-	-	-
		Rolland et al. (2010) **[81]**	Alignement top-down	Variabilité fonctionnelle + Variabilité de coordination + Variabilité d'éxécution	Exigences fonctionnelles et non-fonctionnelles	-	-	-	-
		Mirbel et Crescenzo (2010) **[60]**	-	Variabilité fonctionnelle	Exigences fonctionnelles	-	-	-	-
	i*	*M. Pérez et al. (2010)* **[75]**	Alignement top-down	Variabilité fonctionnelle	Exigences fonctionnelles	Recherche par mots clés et par catégorie	-	-	-
Approches informelles	**Langage naturel**	*K. Zachos et al. 2007* **[102]**	Alignement bottom-up	-	Exigences fonctionnelles	Requêtes XQuery	-	-	Validation empirique (exigences fonctionnelles)
		M. Cremene et al. 2009 **[19]**	Alignement top-down	Variabilité fonctionnelle + Variabilité de coordination	Exigences fonctionnelles	Requêtes à base de mots clés	Similarité sémantique entre les exigences et les description WSDL des services	-	-

Tableau 3.1- Tableau comparatif des différentes approches centrées exigences de composition de services Web

Deuxième partie

Approche et expérimentations

Chapitre 4

Approche multi-perspective centrée exigences de composition de services Web

Dans le chapitre précédent, nous avons mis l'accent principalement sur cinq limitations des approches existantes de composition centrée exigences de services Web qui sont : (i) le traitement partiel de la variabilité, (ii) la négligence des exigences non-fonctionnelles lors des différentes phases du cycle de développement des applications à base de services, (iii) la non-considération des étapes de découverte et de sélection de services logiciels lors de l'alignement de ces derniers avec les exigences des utilisateurs, (iv) la génération en boîte noire du processus de coordinations des services logiciels, et (iv) la validation partielle de la satisfaction des exigences.

Afin de remédier à ces limitations, nous proposons une approche multi-perspective qui positionne la composition des services dans une perspective centrée exigences dans laquelle les services métiers de haut niveau sont décrits en termes d'exigences qu'ils permettent de satisfaire. Un processus d'alignement est proposé pour assurer une mise en correspondance de ces services avec les services logiciels de bas niveau qui sont décrits en termes de déclarations techniques au niveau d'une perspective centrée fonctions. Ce processus d'alignement tient compte des différentes facettes de la variabilité tout au long des étapes de construction des applications à base de services.

Sommaire

4.1 Présentation de l'approche

L'approche que nous proposons dans le cadre de cette thèse permet : (i) la modélisation des applications à base de services en termes d'exigences fonctionnelles et non fonctionnelles ; (ii) la découverte des services logiciels pertinents qui répondent le mieux aux exigences fonctionnelles modélisées lors de la première étape ; (ii) la sélection des services logiciels pertinents et de haute QdS ; (iv) la coordination des services logiciels sélectionnés en vue de les orchestrer ; (v) la validation de la composition afin de vérifier quelle répond aux exigences fonctionnelles et non-fonctionnelles des utilisateurs. Notre approche consiste donc en un processus découpé en cinq étapes successives 1-5 [54] comme le montre la Figure 4.1.

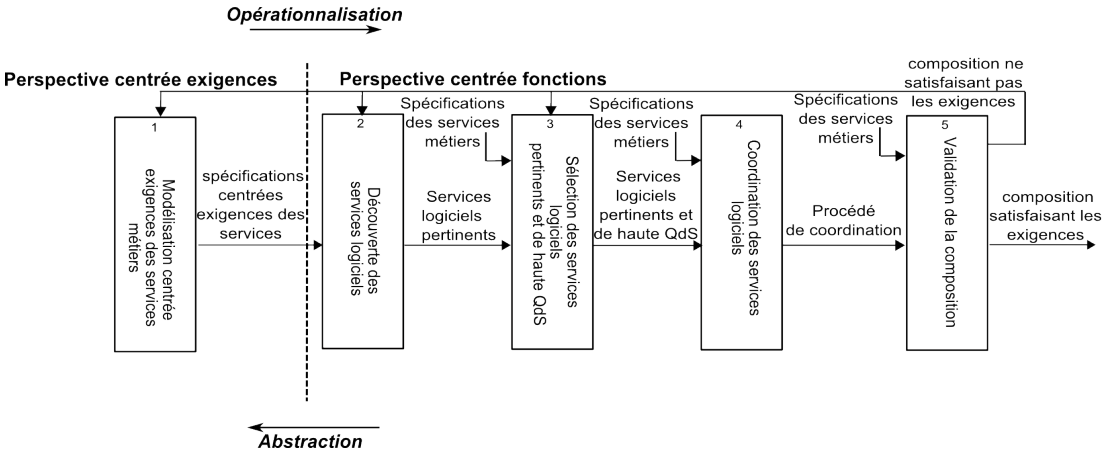


Figure 4.1 — Approche multi-perspective centrée exigences pour la composition de services Web.

La perspective centrée exigences est la perspective où les services à composer sont considérés comme des interactions métiers qui sont décrits en termes d'exigences. Cette perspective inclut la première étape de notre approche :

– **Étape 1. Modélisation centrée exigences des services métiers**. À cette étape, nous proposons une modélisation centrée exigences des services métiers. Cette modélisation consiste à représenter les exigences des utilisateurs et à identifier et spécifier les services métiers et leur composition à partir du modèle d'exigences. La modélisation est réalisée en utilisant le formalisme la Carte [82] qui permet la modélisation des exigences des utilisateurs dans une série de graphes composés d'intentions et de stratégies, appelés cartes. Dans des travaux antérieurs [80] un modèle appelé modèle intentionnel de services (MIS) a été proposé pour spécifier les services présentés par les cartes et qui sont nommés des services intentionnels. Dans ce travail de thèse, le même modèle a été étendu afin d'inclure les aspects de QdS et sera utilisé pour spécifier les services intentionnels.

La perspective centrée fonctions est la perspective où les services logiciels qui réalisent les exigences modélisées sont découverts, sélectionnés et coordonnées. Cette perspective inclut quatre étapes :

- **Étape 2. Découverte des services logiciels pertinents**. À cette étape, nous proposons de chercher les services logiciels pertinents qui répondent le mieux aux exigences fonctionnelles identifiées dans la première étape de l'approche. La recherche est faite par l'interrogation du moteur de recherche de services [1] [11]. L'interrogation se fait en utilisant des mots-clés qui sont extraits à partir des modèles MIS des services intentionnels. Afin de sélectionner plus efficacement les services logiciels pertinents et de haute QdS, nous proposons un filtrage à trois niveaux des services découverts. Dans le premier niveau, nous éliminons les services redondants. Les services qui passent le premier niveau sont filtrés en considérant deux propriétés de QdS à savoir la validité (nous vérifions si les adresses URI des services sont valides) et la disponibilité (nous vérifions si les services sont fonctionnels). Les services qui passent le deuxième niveau sont filtrés selon une mesure de similarité sémantique entre les spécifications intentionnelles fournies par MIS et les spécifications opérationnelles fournies par WSDL. Seuls les services ayant une mesure de similarité supérieure ou égale au seuil empirique considéré sont retenus.

- **Étape 3. Sélection des services logiciels pertinents et de haute QdS**. À cette étape, nous proposons une méthode de guidage automatique pour la sélection des services logiciels pertinents et de haute QdS. Cette méthode consiste à classer l'ensemble des services retenus après l'étape de découverte dans une structure ordonnée appelée treillis de concepts et ceci par l'application de l'Analyse Formelle de Concepts (AFC) [36]. L'AFC est un cadre formel qui permet de regrouper des individus qui partagent des propriétés communes et les organiser en des treillis de concepts. L'AFC permet d'automatiser la tâche de sélection et ceci en fournissant une vue claire et organisée des services afin de permettre aux utilisateurs d'identifier plus facilement les services pertinents et de haute QdS. Notre méthode de guidage intègre également la sélection des services composites en appliquant l'Analyse Relationnelle de Concepts (ARC) qui est une extension de l'AFC. L'ARC permet de grouper les services selon leur degré de composabilité qui est calculé en mesurant la similarité syntaxique et sémantique des opérations et des paramètres d'entrée/sortie des services. La sélection des services composites favorise les services qui requièrent le moins d'adaptation et qui offrent le plus de QdS lors de leur assemblage.

- **Étape 4. Coordination des services logiciels**. À cette étape, nous proposons une génération automatique du procédé de coordination à partir du modèle des exigences élaboré dans la première étape. Pour cela, nous proposons d'utiliser

1. http : //www.service-finder.eu/

la technique de transformation de modèles pour la génération automatique des processus de coordination BPEL à partir des modèles MIS des services intentionnels.

– **Étape 5. Validation de la composition**. À cette étape, nous proposons une validation empirique et une autre automatique des applications élaborées par composition de services. Nous validons empiriquement en termes de précision et de rappel la composition des services par rapport aux exigences fonctionnelles des utilisateurs. La précision évalue le nombre de réels services pertinents et de haute QdS parmi les services sélectionnés et composés par notre approche, tandis que le rappel évalue le nombre de services sélectionnés et composés par notre approche parmi les réels services pertinents et de haute QdS qui existent. Pour les exigences non-fonctionnelles, nous proposons d'utiliser la simulation pour vérifier automatiquement que les services sélectionnés et composés répondent au mieux à ces exigences.

Le reste du chapitre est structuré comme suit. Dans la section 2, nous présentons la modélisation intentionnelle des applications à base de services. Dans la section 3, nous décrivons la découverte des services logiciels effectuée à l'aide de *Service-Finder*. Dans la section 4, nous expliquons comment appliquer l'AFC et l'ARC pour sélectionner les services pertinents et de haute QdS. Dans la section 5, nous présentons les règles de transformation utilisées pour la génération automatique de processus BPEL à partir des modèles MIS des services intentionnels. Dans la section 6, nous démontrons comment valider la satisfaction des exigences fonctionnelles en termes de précision et de rappel. Nous présentons également l'approche de simulation utilisée pour la validation des propriétés de QdS. Finalement, nous terminons par récapituler les contributions apportées par cette thèse.

4.2 Modélisation intentionnelle des applications à base de services

Dans le cadre de ce travail de thèse, nous adoptons une approche centrée exigences qui permet une modélisation de haut niveau d'abstraction des applications à base de services. Cette approche utilise le formalisme la Carte pour représenter les exigences des utilisateurs.

4.2.1 Le formalisme la Carte

Le formalisme la Carte est un système de représentation des processus dans un mode intentionnel. Il fait partie de la classe des modèles de buts. Le système de représentation de la carte utilise le concept d'*intention* et se différencie des autres modèles par l'introduction du concept de *stratégie* pour atteindre un but. Dans

ce système de représentation, une intention est ce qu'on cherche à atteindre. Une stratégie est une manière de réaliser une intention. La Carte permet la modélisation des exigences dans une série de graphes appelés cartes [2]. Une carte un ordonnancement déclaratif et flexible d'intentions et de stratégies.

La Figure 4.2 présente le méta-modèle de la carte, ses concepts clés et leurs relations en utilisant la notation UML.

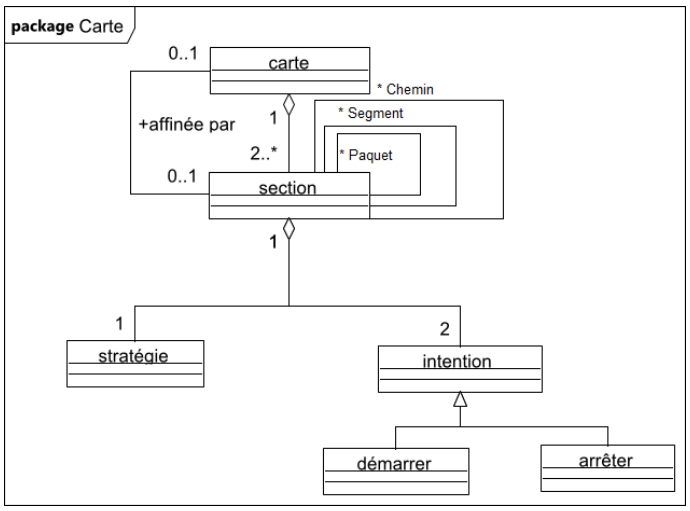


Figure 4.2 — Méta-modèle de la Carte [49].

Une intention représente le but que nous cherchons à atteindre. Selon [47], une intention est une déclaration "optative" qui exprime ce que nous voulons, un état ou un résultat que nous cherchons à atteindre. Chaque carte possède deux buts particuliers : Démarrer et Arrêter pour respectivement commencer et terminer la navigation dans la carte.

De plus, une intention ne peut apparaître qu'une seule fois dans la même carte et chaque intention a les caractéristiques suivantes [49] :

- Elle fait abstraction du processus de sa réalisation ;
- Elle capture "l'essence" du processus ;
- Elle traduit un objectif du métier ;
- Son expression est plus ou moins abstraite.

2. Nous faisons la différence entre Carte qui correspond au formalisme et carte qui correspond à une instance de la Carte.

Afin de normaliser l'écriture des intentions, nous proposons d'utiliser la structure élaborée par Prat [74]. Selon Prat, l'intention est représentée par un verbe suivi d'un ou plusieurs paramètres. Cette structure est présentée à la Figure 4.3.

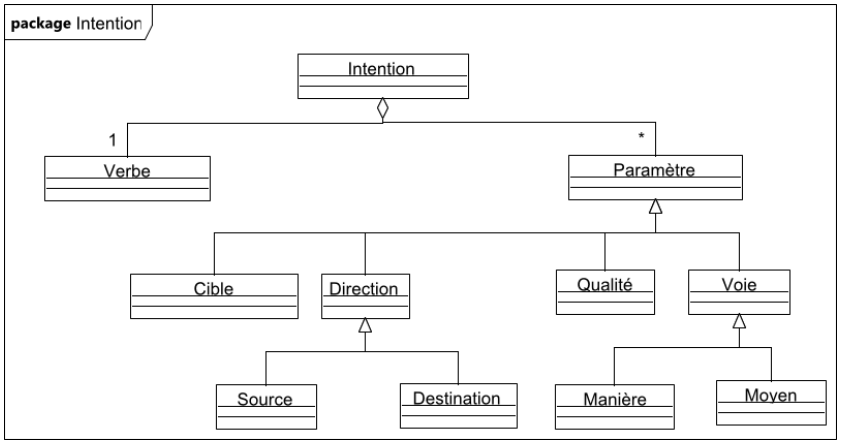


Figure 4.3 — Structure d'une intention [74].

La Figure 4.3 indique qu'une intention est composée d'un verbe suivi d'un ou plusieurs paramètres qui sont :

- Une cible désigne la ou les partie(s) de produit affectée(s) par la réalisation de l'intention. Une cible représente la partie du produit déjà existante sur laquelle l'intention opère un changement ou une partie de produit qui est créée par la satisfaction de l'intention ;

- Une direction pouvant être une source ou une destination. La source montre l'emplacement initial de la partie produit sur laquelle l'intention va opérer. Alors que la destination indique l'emplacement de la partie produit après son utilisation ou sa création ;
- Une qualité définit une propriété de qualité qui contraigne la réalisation de l'intention ;

- Et une voie qui peut être soit un Moyen soit une Manière. Le moyen est un instrument, un outil tandis que la manière, est une approche.

Une telle formulation des intentions permet d'éviter les ambiguïtés du langage naturel qui peuvent conduire à une mauvaise interprétation des intentions de la carte.

Dans une carte, les stratégies correspondent aux différents moyens/manières offerts pour réaliser les intentions. Une stratégie s'associe à l'intention à laquelle elle s'applique. Elle a pour but principal d'extérioriser la façon d'atteindre cette intention puisqu'elle permet de distinguer le but et la façon de le réaliser. En outre,

fournir plusieurs stratégies pour atteindre le même but permet de suggérer des façons alternatives de réaliser ce but. Ceci permet plus d'adaptabilité et de flexibilité dans l'exécution des processus métier modélisés par les cartes.

L'élément principal d'une carte est la section. Chaque section est représentée sous la forme d'un triplet <Ii, Ij, Sij> où :

- Ii est l'intention source ;
- Ii est l'intention cible ;
- Sij est une stratégie permettant de réaliser l'intention Ij à partir de l'intention Ii.

La manière spécifique d'accomplir une intention est capturée dans une section de la carte. Par ailleurs, les diverses sections qui ont les mêmes intentions source et cible définissent différentes stratégies pour réaliser l'intention cible à partir de l'intention source.

Il existe trois relations entre les sections d'une carte [49] :

- **La relation paquet**. Nous avons une relation de type paquet entre plusieurs sections lorsque ces sections ont le même couple d'intentions et que le choix d'une de ces sections pour la réalisation de l'intention cible empêche la sélection des autres sections. Il s'agit de plusieurs sections mutuellement exclusives. En d'autres termes la relation entre ces sections est de type OU Exclusif. Un paquet est représenté graphiquement par une flèche en pointillés. "$\otimes$" est l'opérateur utilisé pour relier deux ou plusieurs sections exclusives.
- **La relation segment**. Dans une carte, il est possible de réaliser une intention cible à partir d'un but source en utilisant plusieurs stratégies complémentaires. Chacune de ces stratégies, couplée avec l'intention source et le but cible, définit une section dans la carte. Cette topologie est appelée multi-segment. Elle peut être vue comme une relation logique ET/OU. Les sections appartenant à un multi-segment possèdent le même but source et le même but cible. "$\vee$" est l'opérateur utilisé pour relier deux ou plusieurs sections complémentaires.
- **La relation chemin**. La relation entre les sections d'une carte établit quelles intentions précèdent ou succèdent telles autres. La relation de précédence indique qu'une intention ne peut être réalisée que si une autre intention a été réalisée. La relation de précédence conduit à ordonner les sections dans un chemin. Un chemin est donc un sous-ensemble de sections de la carte où les sections sont reliées par une relation de précédence. La relation de type chemin est notée par "$\bullet$".

Enfin, il est possible d'affiner une section par une carte entière. L'affinement est un mécanisme d'abstraction par lequel un assemblage complexe de sections au niveau i+1 est considéré comme une section unique au niveau i.

Nous avons choisi d'utiliser le formalisme la Carte, premièrement parce que ce formalisme a été déjà utilisé pour la modélisation des applications à base de services Web [80], [81] et [60] donc nous pouvons réutiliser les connaissances antérieures, et deuxièmement parce que nous voulons bénéficier des atouts de ce formalisme à savoir :

- **La modélisation explicite de l'intentionnalité**. La Carte modélise explicitement l'aspect intentionnel des applications à base de services. Le point de vue intentionnel d'une application à base de services facilite la découverte et la sélection de services. En effet, à partir de ce qu'il souhaite, l'utilisateur peut identifier les services qui répondent le mieux à ses exigences ;

- **La simplicité du langage**. La Carte utilise un langage simple et facile à comprendre par des utilisateurs non experts du domaine des services web à la différence des langages de services qui exigent des connaissances techniques approfondies ;

- **L'expression de la variabilité**. La Carte donne une multitude de chemins entre deux intentions. Chaque chemin correspond à une variante d'usage de l'application à base de services ;

- **La notion d'affinement**. La Carte représente une application à base de services à différents niveaux de granularité. Cette notion d'affinement offre une certaine modularité. En effet, chaque carte d'affinement peut être vue comme une composition à part qui peut être réutilisée par d'autres applications.

Dans ce qui suit, nous présentons le modèle MIS permettant la spécification des services intentionnels et nous expliquons les directives à suivre pour identifier les services intentionnels et leur composition à partir des cartes.

4.2.2 Le modèle intentionnel de services MIS

Les services intentionnels sont des services présentés par les cartes. Ils permettent la réalisation des exigences des utilisateurs modélisées sous forme d'intentions à l'aide du formalisme la Carte. Les services intentionnels sont spécifiés par le modèle intentionnel de services MIS. La Figure 4.4 présente le méta-modèle de MIS en utilisant les notations du diagramme de classes UML.

La Figure 4.4 montre que la spécification d'un service intentionnel comporte quatre parties : l'interface, le comportement, la composition et la QdS. Nous décrivons

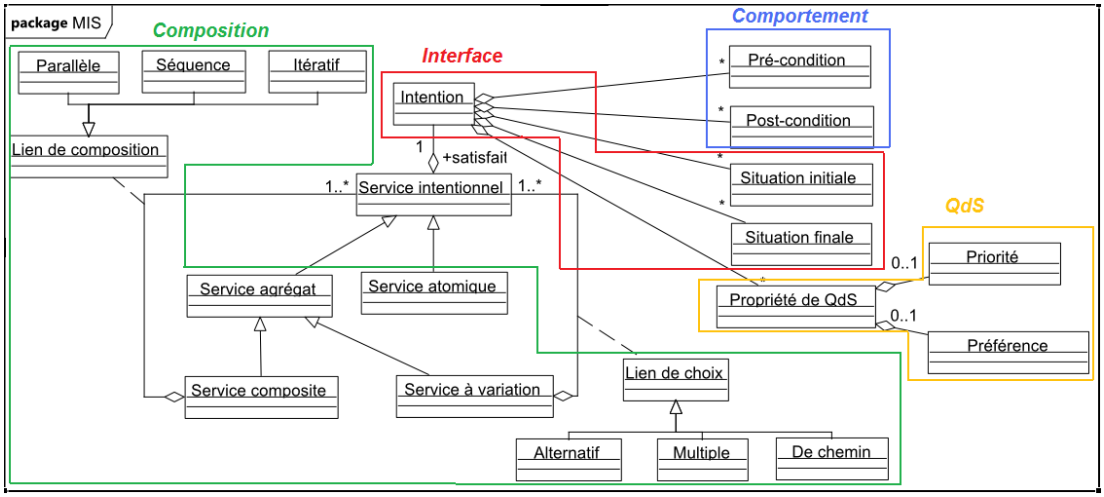


Figure 4.4 — Le méta-modèle de MIS.

chacune de ces parties dans les paragraphes suivants.

- **L'interface**. Il y a trois éléments décrivant la partie interface qui sont : l'*Intention*, la *Situation initiale* et la *Situation finale*. L'*Intention* représente l'identité du service intentionnel, la *Situation initiale* définit les paramètres d'entrée et la *Situation finale* fournit les paramètres de sortie.

- **Le comportement.** Il y a deux éléments décrivant la partie comportement d'un modèle intentionnel d'un service qui sont : la *Pré-condition* et la *Post-condition*. La *Pré-condition* et la *Post-condition* sont respectivement l'état initial et final, à savoir, l'état nécessitant la réalisation de l'intention et l'état résultant de sa réalisation.

- **La composition**. Il y a deux types de services : un service *atomique* et un service *agrégat*. Un service *atomique* n'est pas décomposable en d'autres services intentionnels, alors qu'un service *agrégat* l'est. Un service *atomique* est associé à une intention dite "opérationnalisable", alors qu'un service *agrégat* est associé à une intention de haut niveau qui doit être décomposable afin d'atteindre des sous-intentions opérationnalisables. Une intention opérationnalisable est une intention pour laquelle il est possible de définir une série d'opérations permettant sa réalisation. Dans ce sens, nous pouvons dire qu'un service *atomique* est directement exécutable, alors qu'un service agrégat ne l'est pas. Nous distinguons deux types de services agrégats : ceux qui sont opérationnalisables par une décomposition OU de l'intention, les variants et ceux qui sont opérationnalisables par une décomposition ET, les composites. Il existe trois types de variation : *choix alternatif*, *choix multiple* et *choix de chemin*. Le *choix alternatif* correspond à une relation OU_{ex} entre services (nous associons au *choix alternatif* le symbole

"⊗"). Le *choix multiple* correspond à une relation OU entre services (nous associons au *choix multiple* le symbole "∨"). Finalement, le *choix de chemin* correspond à des enchaînements alternatifs de services (nous associons au *choix de chemin* le symbole "∪"). Il existe aussi trois types de composition : la séquence, le parallélisme et l'itération. Dans une *séquence*, les services composants sont exécutés d'une manière séquentielle (nous associons à la *séquence* le symbole "•"). Dans un *parallélisme*, les services composants sont exécutés d'une manière parallèle (nous associons au *parallélisme* le symbole "//"). Dans une *itération*, les services composants sont exécutés à plusieurs reprises (nous associons à l'*itération* le symbole "∗").

– **La QdS**. Il y a trois éléments décrivant la partie QdS d'un modèle intentionnel d'un service qui sont : la *Propriété de QdS*, la *Priorité* et la *Préférence*. La *Propriété de QdS* exprime un critère de QdS comme le temps de réponse ou la disponibilité. Par la *Priorité*, nous associons une priorité à la *Propriété de QdS*, et par la *Préférence* nous exprimons une préférence.

4.2.3 Identification des services intentionnels

Afin d'identifier les services intentionnels et leur composition à partir d'une carte, nous proposons de suivre les trois directives suivantes [80] :

1. **Identification des services atomiques** : nous utilisons une seule règle qui est la suivante :
   R : un service atomique est associé à chaque section opérationnalisable d'une carte. Une section opérationnalisable est une section qui ne peut être affinée par une autre carte.

2. **Identification de tous les chemins d'une carte** : pour identifier tous les chemins qui existent entre l'intention Démarrer et l'intention Arrêter, nous appliquons un algorithme permettant d'identifier tous les chemins dans un automate fini. Cet algorithme est une adaptation de l'algorithme de R. MacNaughton et H. Yamada [55]. L'algorithme de de R. MacNaughton et H. Yamada permet d'identifier tous les chemins qui existent entre un nœud initial et un nœud final qui présentent respectivement, dans le cas de la carte, l'intention démarrer et l'intention arrêter. Cet algorithme se base sur les deux formules suivantes :

$$Y_{s,Q,C} = \bullet(X^*_{s,Q\setminus\{s\},s}, X_{s,Q\setminus\{s,C\},C}, X^*_{t,Q\setminus\{s,t\},C}) \tag{4.1}$$

$$X_{s,P,C} = \cup((X_{s,P\setminus\{p\},p}), \bullet(X_{s,P\setminus\{p\},p}, X^*_{s,P\setminus\{p\},p}, X_{s,P\setminus\{p\},c})) \\ si\ \mathrm{P} = \emptyset \text{ alors } X_{s,p,t} = X_{s,t} \tag{4.2}$$

où :
s, c : les nœuds source et cible désignant respectivement l'intention démarrer et l'intention arrêter.
Q : l'ensemble des intentions intermédiaires incluant les intentions s et c.
P : l'ensemble des intentions intermédiaires entre s et c et n'incluant pas ces derniers.
p : est une intention quelconque de l'ensemble P.
• : opérateur de composition.
$*$: opérateur d'itération.

La formule (4.1) de $Y_{s,Q,c}$ indique tous les chemins possibles entre l'intention source s et l'intention cible c et qui sont : les chemins de l'intention s vers l'intention s qui passant par l'ensemble Q des intentions intermédiaires sans inclure l'intention s, suivies de tous les chemins entre s et c sans passer par s et c et tous les chemins de l'intention finale c vers l'intention finale c sans passer par les intentions source et cible s et c.

La formule (4.2) de $Y_{s,P,c}$ indique tous les chemins entre l'intention source s et l'intention cible c en passant par l'intention intermédiaire p. $Y_{s,P,c}$ est l'union des chemins entre s et c sans passer par l'intention p ainsi que les chemins qui passent par p.

3. **Identification des services agrégats** : nous utilisons les règles suivantes afin d'identifier les différents types de services agrégats à partir des sections d'une carte et les relations qui existent entre elles :

   – R1 : Affecter à chaque paquet de la carte un service à choix alternatif.

   – R2 : Affecter à chaque multi-segment de la carte un service à choix multiple.

   – R3 : Affecter à chaque chemin de la carte un service composite séquentiel.

   – R4 : Affecter à chaque multi-chemin de la carte un service à variation de chemin.

   Les règles d'identification des services logiciels atomiques et agrégats et qui sont : R, R1, R2, R3 et R4 s'appliquent sur tous les niveaux de la carte en cas où nous avons des sections affinées par d'autres cartes.

4.3 Découverte des services logiciels

Pour découvrir les services logiciels permettant l'exécution des services intentionnels représentés par les cartes et spécifiés par les modèles MIS, nous interrogeons le

moteur de recherche de services *Service-Finder*. *Service-Finder* est un environnement web 2.0 dédié à la découverte de services, il permet la recherche dans plus de 25 000 services web liés à plus de 200 000 pages Web [11]. Nous avons choisi d'utiliser *Service-Finder* parce qu'il fournit pour chaque service découvert des informations sur sa QdS, plus précisément sur sa disponibilité et son temps de réponse. Ces informations sont obtenues par monitoring.

L'interrogation de *Service-Finder* se fait par mots-clés. Ces mots-clés sont extraits à partir des modèles intentionnels des services métiers. L'extraction des mots clés est réalisée moyennant la formule TF-IDF (Term Frequency- Inverse Document Frequency) [85]. TF-IDF est une méthode de pondération souvent utilisée en recherche d'information ou encore en fouille de données. C'est une mesure statistique qui permet d'évaluer l'importance d'un terme contenu dans un document, relativement à une collection ou un corpus. Le poids du terme augmente proportionnellement au nombre de ses occurrences dans le document. Il varie également en fonction de la fréquence du mot dans le corpus. La fréquence d'occurrence d'un terme t dans un document d est calculée par la formule TF :

$$TF = F(t,d)/Max[F(t,d)]$$

où Max [f(t,d)] est la fréquence maximale des termes dans d.
La fréquence inverse de document est donnée par la formule IDF :

$$IDF = log(N/n)$$

où N est le nombre de documents dans le corpus, et n ceux qui contient le terme t.
Une formule de TF*IDF est donc la multiplication de la formule TF par la formule IDF :

$$TF - IDF = [F(t,d)/Max[F(t,d)]] * log(N/n)$$

Une formule TF-IDF combine donc les deux critères : 1. L'importance du terme pour un document (par la mesure TF), et 2. Le pouvoir de discrimination de ce terme (par la mesure IDF). Ainsi, un terme qui a une valeur de TF*IDF élevée doit être à la fois important dans ce document, et aussi il doit apparaître peu dans les autres documents. Pour notre cas, notre corpus est l'ensemble des spécifications MIS des services métiers. Un terme est considéré comme mot-clé d'un service s'il a un nombre d'occurrence important dans la spécification MIS du service et s'il apparaît peu dans les spécifications MIS des autres services MIS. Nous proposons aussi d'appliquer la mesure TF-IDF sur les spécifications des services métiers après avoir éliminer les mots vides de sens. Pour ce faire, nous utilisons une liste appelée stoplist contenant tous les mots que nous ne voulons pas garder comme les prépositions, les adverbes, les

prénoms, les adjectifs etc. Donc, quand nous rencontrons un mot dans un document, nous devons vérifier s'il apparaît à cette liste. Si c'est le cas, nous procédons à son élimination.

Après avoir invoqué le moteur de recherche, et pour réduire l'ensemble des services retournés, nous procédons à un filtrage à trois niveaux [28]. Au premier niveau, nous éliminons les services redondants. Si un service apparaît plusieurs fois dans l'ensemble de services fourni par Service-Finder suite à son interrogation, alors nous laissons uniquement sa première occurrence. Nous considérons que deux services sont identiques s'ils ont la même interface d'invocation, c'est-à-dire la même adresse URI. Au deuxième niveau, nous examinons les services qui restent selon deux propriétés de QdS à savoir la validité et la disponibilité. Ces deux propriétés sont vérifiées comme suit :

- la validité : nous vérifions si l'adresse URI figurant au niveau de la spécification WSDL du service est valide. Un message du type "adresse invalide" est affiché si l'adresse URI d'un service est invalide ;

- la disponibilité : nous vérifions si le service est fonctionnel ; c'est-à-dire s'il fournit une réponse suite à son invocation. Un message du type "le service est momentanément indisponible" est affiché si un service ne répond pas suite à son invocation.

Le troisième niveau de filtrage est basé sur une mesure de similarité sémantique entre les modèles MIS des services intentionnels et les spécifications WSDL des services logiciels. Seuls les services opérationnels dont les spécifications WSDL ont une valeur de similarité sémantique supérieure ou égale à un seuil empirique considéré sont retenus. La mesure de la similarité sémantique est calculée moyennant la formule de G. Pirrò et N. Seco [71]. Cette formule est implémentée et disponible au niveau de la librairie JWSL (Java WordNet Similarity Library) [3]. Le choix de cette librairie est justifié par son utilisation de la base WordNet [4]qui offre l'avantage d'une grande couverture lexicale. La mesure de G. Pirrò et N. Seco est basée sur la mesure de similarité de Resnik [78] qui permet de calculer le contenu informatif d'un concept de la base WordNet. La mesure de G. Pirrò et N. Seco utilise donc la mesure de Resnik pour calculer la similarité sémantique entre deux concepts tout en intégrant également la valeur d'abstraction spécifique et en commun entre ces deux concepts. La mesure de G. Pirrò et N. Seco (PISE) entre les deux concepts s et t est la suivante :

$$PISE(s,t) = \begin{cases} 3 * IC(msca(s,t)) - IC(s) - IC(t) \text{ si } s \neq t \\ 1 \text{ si } s = t \end{cases}$$

3. http ://grid.deis.unical.it/similarity/
4. http ://wordnet.princeton.edu/

Avec : $IC(w) = -logP(w)$ où w est un mot et P(w) est la probabilité de trouver le mot w dans la base wordnet. IC (Information Content) est la mesure de Resnik permettant de calculer le contenu informatif d'un mot. La mesure msca (most specific common abstraction) permet de calculer la valeur d'abstraction spécifique et en commun entre deux mots de la base WordNet.

Pour évaluer la similarité sémantique entre les spécifications MIS des services métiers et spécifications WSDL des services logiciels, nous procédons par caluler la similarité entre les noms des opérations (OpSim) et par calculer la similarité entre les paramètres d'entrée (InParSim) et les paramètres de sortie (OutParSim). Le résultat final est le produit de OpSim, InParSim et OutParSim.

4.4 Sélection des services logiciels

Pour automatiser la sélection des services pertinents et de haute QdS, nous proposons d'utiliser l'Analyse Formelle de Concepts (AFC) et son extension l'Analyse Relationnelle de Concepts [36]. L'AFC a été utilisée dans plusieurs travaux qui relatent à la classification des services Web comme [70], [4], [6], [15] et [33]. Le plus de notre travail est l'application de ce cadre de travail pour la sélection des services Web. Cette sélection tient compte des propriétés fonctionnelles et non-fonctionnelles des services.

4.4.1 Introduction à l'AFC

L'AFC [84] est une théorie basée sur la théorie des treillis [10] et des treillis de Galois [7], qui permet de regrouper des individus qui partagent des propriétés communes. Les groupes d'individus et de propriétés mutuellement correspondants sont appelés des *concepts formels* [10]. Ces concepts sont organisés en une hiérarchie, dite *treillis de concepts*, par le biais de la relation d'ordre partielle qui repose sur la relation d'inclusion ensembliste entre groupes d'individus et de propriétés.

Un concept formel est constitué de deux parties : l'extension qui contient les individus appartenant au concept et l'intension qui contient les propriétés partagées par les individus. Ces concepts sont organisés en une hiérarchie, dite treillis de concepts, suivant une relation d'incidence binaire entre individus et propriétés. Dans l'AFC, les données sont représentées sous la forme d'un tableau booléen à deux dimensions, appelé contexte formel, définissant la relation d'incidence binaire entre deux ensembles finis d'individus et de propriétés. Un contexte formel se décrit formellement comme suit :

Un contexte formel est un triplet K = (O,A,I) où :

– O est l'ensemble des individus ;

– A est l'ensemble des propriétés ;

– I est une relation d'incidence binaire entre les éléments de O et les éléments de A telle que I $\subseteq$ O $\times$ A indiquant pour chaque individu les propriétés qui lui sont associées.

Un contexte peut être représenté graphiquement par un tableau de dimension |O| $\times$ |A|. Un exemple d'un contexte formel est représenté par le Tableau 4.1. Dans ce tableau, des chercheurs en génie logiciel (les individus) sont décrits par les thématiques de recherche et les publications (les propriétés).

	Thématique de recherche			Publication			
	Ingénierie des modèles(IM)	Ingénierie des exigences(IE)	Ingénierie des services(IS)	Journal of Artificial Intelligence Research (JAIR)	Journal of Computer Security (JCS)	Distributed Systems Enginee-ring (DSE)	Information Systems Research (ISR)
Anne	×			×			×
Michel	×		×		×		
Jean			×	×		×	
Marie		×					×
Paul	×	×					
Emilie	×		×				×
Pierre	×		×		×		

Tableau 4.1: Exemple d'un contexte formel décrivant les chercheurs en génie logiciel par leurs thématiques de recherche et leurs publications.

Pour chaque ensemble d'objets X $\in$ P(O), où P représente l'ensemble des parties d'un ensemble X, les propriétés partagées par ces individus sont obtenues à l'aide de l'application f : P($\mathcal{O}$)$\rightarrow$ P(A) définie par f(X) = X'= {a $\in$ A | $\forall$ o $\in$ X, (o, a) $\in$ I}. Symétriquement, l'application g : P(A)$\rightarrow$P(O), définie par g(Y) = Y'= {o $\in$ O | $\forall$ a $\in$ Y, (o, a) $\in$ I}, associe à un ensemble de propriétés tous les individus partagés par ces propriétés.

Dans le contexte illustré par le Tableau 4.1, nous avons f ({Anne, Michel})= IM et g ({IM})= {Anne, Michel, Paul, Emilie, Pierre}.

Un concept formel est constitué de deux parties se caractérisant mutuellement : l'extension qui contient les individus appartenant au concept et l'intension qui contient les propriétés partagées par les individus. Un concept formel se décrit formellement comme suit : Un concept formel d'un contexte K = (O, A, I), est une paire (X, Y) où :

– X $\in$ P(O) est appelé l'extension ;

– Y ∈ P(A) est appelé l'intension ;

– Une paire (X,Y) est un concept formel de K si et seulement si : X ⊆ O, Y ⊆ A, X = Y' et Y = X'.

Les concepts d'un contexte donné sont ordonnés naturellement par une relation de *sous-concept* et de *super-concept* [36] qui repose sur la relation d'inclusion entre des sous ensembles d'individus et de propriétés : (X1, Y1)≤ (X2, Y2) :⇔ X1 ⊆ X2(⇔Y1 ⊆ Y2). L'ensemble de tous les concepts formels extraits du contexte K = (O, A, I), ordonné par la relation de super-concept (ou sous-concept), est désigné par L = $< C_K, \leq_K >$ et est appelé un treillis de concepts de K. La relation d'ordre partiel $\leq_L$ entre les concepts correspond à l'inclusion des extensions (ou l'inclusion inverse des intensions).

Le treillis de la Figure 4.5 correspond au contexte formel donné dans le Tableau 4.1.

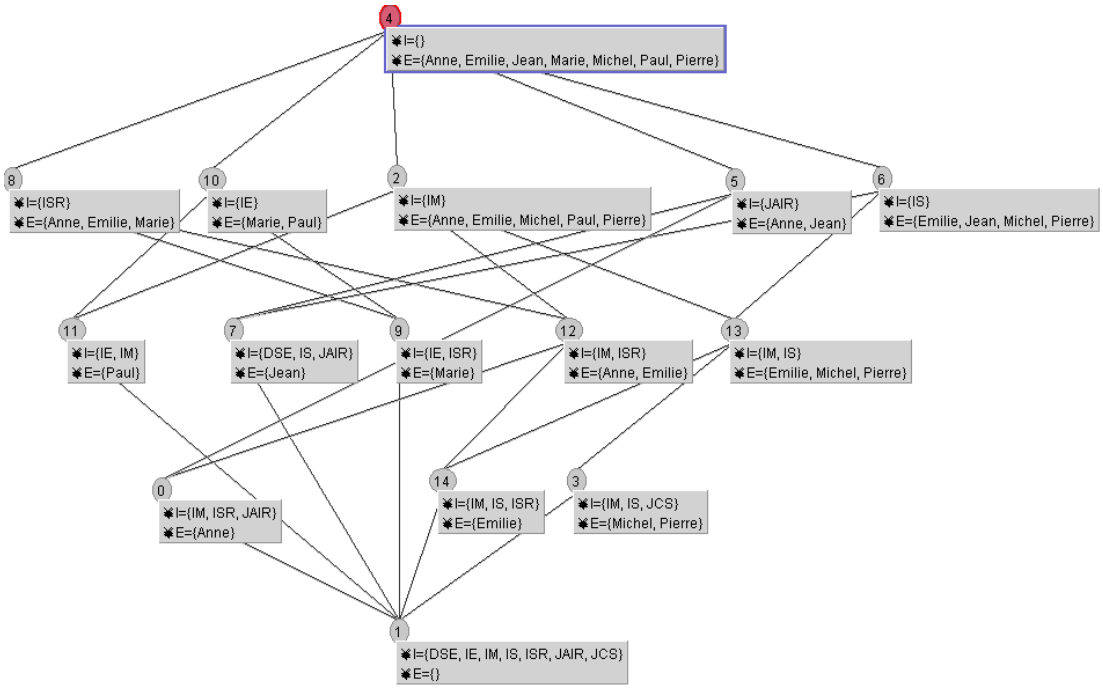


Figure 4.5 — Treillis avec étiquetage complet construit à partir du contexte formel présenté par le Tableau 4.1.

Le nœud 2 du treillis contenant le couple ({Anne, Emilie, Michel, Paul, Pierre}, {IM}) correspond au concept c2 du treillis : les individus Anne, Emilie, Michel, Paul et Pierre travaillent sur la même thématique de recherche qui est l'ingénierie des modèle (IM) qui, à son tour, caractérise exclusivement ces cinq individus. La Figure 4.5 présente un treillis avec *étiquetage complet*, c'est-à-dire que tous les individus et toutes

les propriétés qui définissent un concept sont explicitement indiqués dans l'intension et l'extension du concept. Cet affichage complet peut devenir rapidement gênant, surtout quand les nombres d'individus et d'attributs sont élevés. Pour éviter ce problème, il est possible d'utiliser un affichage plus compact des treillis grâce aux extensions et intensions simplifiées des concepts. L'extension (respectivement l'intension) simplifiée d'un concept (X2, Y2) est l'ensemble X1 $\subseteq$ X2 (respectivement X1 $\subseteq$ X2) des individus (respectivement des propriétés) du concept qui n'apparaissent pas dans l'extension de ses sous-concepts (respectivement dans l'intension de ses super-concepts).

Nous nommerons treillis avec *étiquetage réduit* les représentations de treillis comportant l'intension et l'extension simplifiées des concepts. La Figure 4.6 présente un treillis avec étiquetage réduit du contexte présenté par le Tableau 4.1.

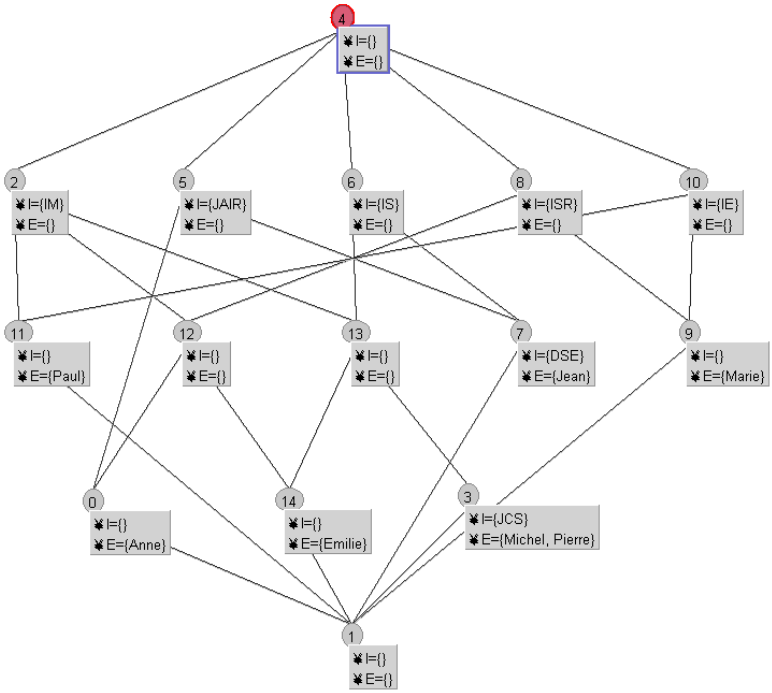


Figure 4.6 — Treillis avec étiquetage réduit construit à partir du contexte formel présenté par le Tableau 4.1.

Dans le concept 2 du treillis, nous avons tous les chercheurs qui travaillent sur la thématique IM. L'extension complète de ce concept est obtenue en cumulant sans répétition tous les individus se trouvant en-dessous de ce concept à savoir les concepts : 11, 12, 13, 0, 14, 3 et 1. Ceci nous permet de déduire qu'Anne, Emilie, Michel, Paul et Pierre sont les chercheurs qui travaillent sur IM.

Nous pouvons voir que ce treillis de la Figure 4.6 est plus compact (il y a moins du

texte au niveau des intensions et des extensions des concepts du treillis) que celui de la Figure 4.5. Les treillis à étiquetage réduit peuvent toutefois devenir moins pratiques quand le nombre de concepts est élevé, car il faut alors naviguer dans le treillis pour connaître tous les éléments situés dans l'extension et l'intension d'un concept donné. Dans le reste de ce manuscrit, sauf mention contraire, les treillis à étiquetage réduit seront systématiquement utilisés.

4.4.2 Utilisation de l'AFC pour la sélection des services pertinents et de haute QdS

Nous avons choisi d'utiliser l'AFC parce qu'elle nous permet de sélectionner et de composer dynamiquement les services web. En effet, en réponse à des exigences de QdS, l'AFC fournit non pas un seul service présentant la solution optimale, mais un ensemble formé d'un ou de plusieurs services. Ces services partagent des propriétés communes de QdS et présentent ainsi des candidats de substitution qui peuvent être interchangés dynamiquement pour pallier des problèmes de pannes ou d'indisponibilités rencontrés lors d'une composition de services.

Pour notre problème de sélection de services, nous définissons un contexte K où les individus sont les services pertinents obtenus après l'étape de découverte et les propriétés représentent des propriétés de QdS [28]. Nous considérons deux propriétés de QdS : la disponibilité et le temps de réponse. Les valeurs de la disponibilité et du temps de réponse des services sont fournies par Service-Finder. La relation d'incidence binaire est : un service "est caractérisé par" une propriété de QdS. Par l'application de l'AFC et en considérant le contexte K, nous voulons identifier les services pertinents et qui offrent le meilleur compromis entre disponibilité et temps de réponse.
Les valeurs de la disponibilité et du temps de réponse sont données en deux unités différentes (la disponibilité est en % et le temps de réponse est en ms). Pour cette raison, nous proposons d'échelonner les valeurs de ces deux propriétés de QdS afin de faire correspondre pour chaque échelon une valeur qualitative ordinale. Pour l'échelonnage, nous utilisons la technique statistique du boxplot [14]. Un boxplot divise un ensemble de valeurs numériques en quatre quarts, ou quartiles. La Figure 4.7 montre un exemple typique d'un boxplot.

Le quartile inférieur Q_i d'un ensemble de valeurs est la valeur telle que 25% des valeurs sont égales ou inférieures à cette valeur et il représente le 25^e percentile. Le quartile supérieur Q_s représente le 75^e percentile. Les quartiles inférieur et supérieur de la distribution des valeurs représentent les limites du boxplot. La distance interquartile du boxplot, IQ = Qs - Qi, correspond à la distance entre le quartile inférieur et le quartile supérieur. La distance interquartile est utilisée pour calculer les queues de la distribution. Les valeurs des queues inférieure et supérieure sont calculées respectivement par Q_i - 1,5× IQ et Q_s + 1,5 × IQ.

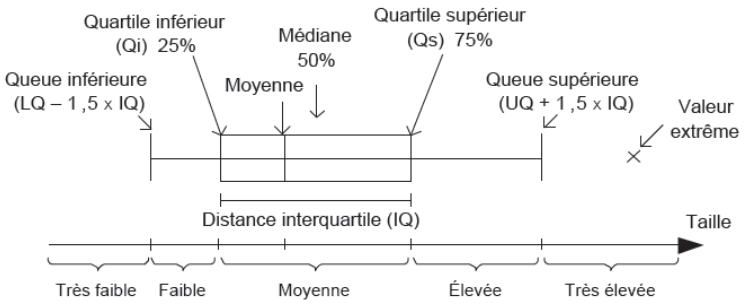


Figure 4.7 — Le Boxplot.

Nous associons des valeurs qualitatives ordinales définies avec une échelle de Likert à 5 points [57] aux quartiles du boxplot comme suit : très faible (très élevé, respectivement) correspond aux valeurs de la disponibilité et du temps de réponse en dessous de la queue inférieure (au-dessus de la queue supérieure, respectivement) ; faible (élevé, respectivement) correspond aux valeurs entre la queue inférieure et le quartile inférieur (entre la queue supérieure et le quartile supérieur, respectivement) ; moyen correspond aux valeurs comprises entre les quartiles inférieur et supérieur.

4.4.3 Introduction à l'ARC

L'ARC [45] [41] est une extension de l'AFC permettant l'extraction de concepts formels à partir d'ensembles d'individus décrits par des propriétés et des liens inter-individus. Contrairement à l'AFC, l'ARC opère sur un ensemble de contextes. Cet ensemble de contextes se nomme famille de contextes relationnels (FCR). Les contextes qui composent une FCR sont de deux types :

- Contexte formel : identique aux contextes de l'AFC, qui relie des individus à des propriétés,
- Contexte relationnel : contexte qui relie des individus d'un contexte formel à des individus d'un contexte formel.

Les concepts formés sont dits *concepts relationnels* car les intensions qu'ils renferment font référence à d'autres concepts.

Formellement, une FCR est une paire *(K, R)* où :

- K est un ensemble de contextes formels $K_i = (O_i, A_i, I_i)$;
- R est un ensemble de relations binaires $r_k \subseteq O_i \times O_j$, où O_i et O_j sont des ensembles d'individus de contextes formels K_i et K_j appelés respectivement

domaine et co-domaine de r_k.

Considérons la relation *Collaboration* qui définit une relation de collaboration entre les différents chercheurs. Le contexte formel du Tableau 4.1 et le contexte relationnel du Tableau 4.2 forment un échantillon FCR.

	Anne	Michel	Jean	Marie	Paul	Emilie	Pierre
Anne			×				
Michel						×	×
Jean							
Marie					×		
Paul							
Emilie							
Pierre							

Tableau 4.2 — Contexte relationnel décrivant les chercheurs qui collaborent ensemble.

Les contextes de la FCR sont traités à par les algorithmes classiques de dérivation de structures conceptuelles de l'AFC. Ainsi, à partir d'une FCR, nous obtenons un ensemble de treillis, un par contexte, appelé *Famille de Treillis Relationnels* (FTR). La construction de FTR est un processus itératif. Lors d'une étape initiale, un treillis L_i^0 est construit pour chaque contexte formel $K_i \in K$. Les concepts de ces treillis sont ensuite utilisés pour enrichir les contextes relationnels $R_j \in R$. À l'aide des contextes enrichis, les différents treillis précédemment construits sont mis à jour. Les concepts de ces treillis sont de nouveau utilisés pour mettre à jour les contextes relationnels, et ainsi de suite jusqu'à obtention d'un point fixe. Le point fixe est atteint lorsque les contextes génèrent les mêmes treillis qu'à l'étape précédente.

Le treillis final correspondant à notre exemple est présenté sur la Figure 4.8. L'individu Anne dans le concept c0 a une relation collaboration avec Jean qui apparaît dans l'extension du concept c7. Anne et Jean partagent la propriété commune qui est JAIR apparaissant dans l'intension du concept partagé c5. L'individu Michel est assigné à la propriété relationnelle collaboration :c13, ce qui signifie que Michel a une relation commune collaboration avec les individus situés dans l'extension des sous-concepts de c13, en l'occurrence Emilie et Pierre. Michel, Emilie et Pierre partagent la propriété commune IM se trouvant dans l'intension du concept c2. Enfin, l'individu Marie entretient une relation de *collaboration* avec l'individu Paul qui se trouve dans l'extension du concept c11. La propriété partagée par Marie et Paul est IE. Nous pouvons observer, grâce aux concepts générés par ARC, que des individus travaillant sur la même thématique de recherche ou qui publient dans les mêmes journaux entretiennent une relation de collaboration, chose que nous n'aurions pas pu voir autrement.

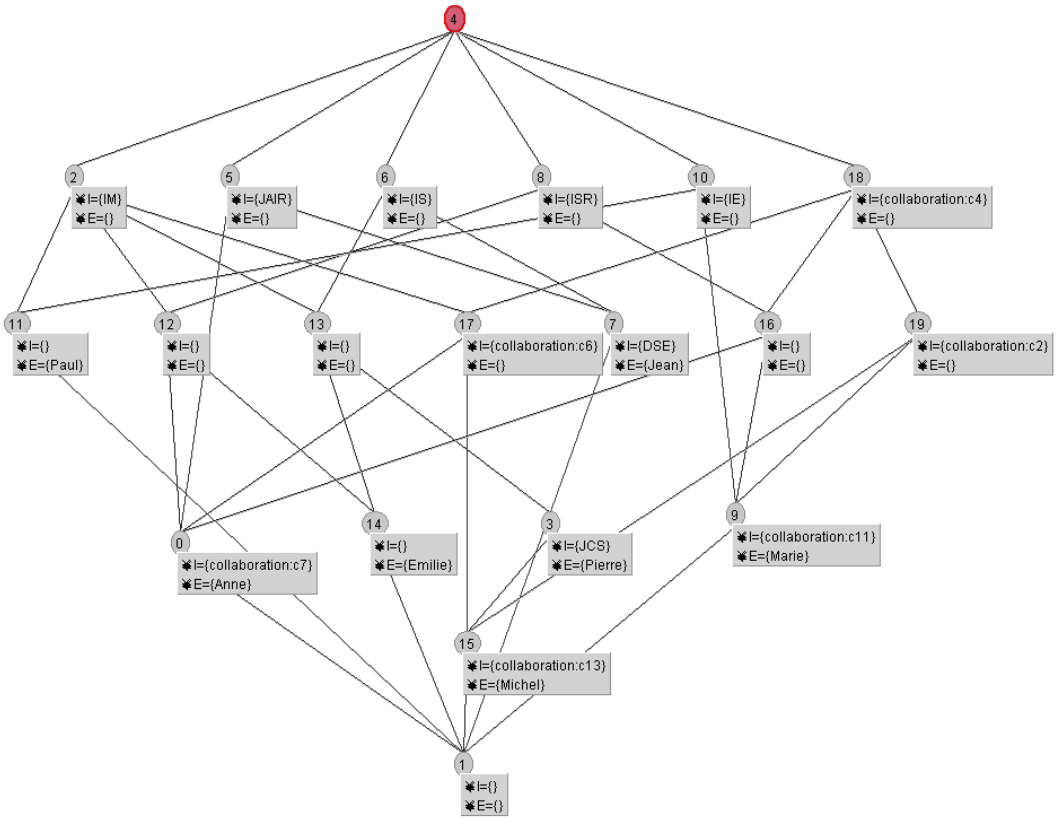


Figure 4.8 — Treillis final de l'exemple obtenu par application de l'ARC.

4.4.4 Utilisation de l'ARC pour la sélection des services composites pertinents et de haute QdS

Un service intentionnel peut être vu comme séquence composée de deux ou plusieurs opérations élémentaires dont l'exécution de chacune dépend de l'exécution de l'opération qui la précède en termes de paramètres d'entrée/sortie. Chacune de ces opérations peut être réalisée par un service logiciel atomique. La découverte de ces services est faite comme expliqué précédemment sauf que nous proposons d'appliquer un filtrage syntaxique au lieu du filtrage sémantique. Ce filtrage syntaxique permet d'examiner les signatures des services pour en garder que ceux qui satisfont l'exigence requis. Pour ce faire, nous définissons trois degrés de compatibilité syntaxiques examinant les signatures des opérations [5] :

- Compatible : si le service contient au moins une opération qui satisfait l'exigence requis.

- Adaptable : si le service nécessite l'adaptation de son opération ou de ses paramètres d'entrée et/ou de sortie pour satisfaire l'exigence requis.

– Incompatible : si le service ne contient aucune opération qui satisfait l'exigence requis.

Ces degrés de compatibilité sont mesurés à l'aide de la métrique proposée dans [17]. Cette dernière permet de calculer la distance syntaxique qui existe entre deux chaînes de caractères. Le choix de cette métrique est justifiée par les expérimentations présentées dans [17] et qui ont prouvé son efficacité par rapport à d'autres métriques. Le schéma de cette métrique est hybride et il combine la distance TF-IDF [18] et la distance Jaro-Winkler [48].

Les services obtenus sont alors classifiés selon les quatre modes de composition suivantsé [5] :

– Entièrement composable : lorsque l'opération du service source couvre par ses paramètres de sortie tous les paramètres d'entrée de l'opération du service cible.

– Partiellement composable : lorsque un ou plusieurs paramètres d'entrée de l'opération cible ne sont pas couverts.

– Adaptable-Entièrement composable : lorsque l'opération du service source et l'opération du service cible sont entièrement composables, mais des adaptations sont nécessaires pour les paramètres de sortie de l'opération source et/ou pour les paramètres d'entrée de l'opération cible.

– Adaptable-partiellement composable : lorsque l'opération du service source et l'opération du service cible sont partiellement composables, mais des adaptations sont nécessaires pour les paramètres de sortie de l'opération source et/ou pour les paramètres d'entrée de l'opération cible.

La sélection des services composites est procédée en considérant des familles relationnelles par paire de services s'exécutant séquentiellement. Chaque famille renferme des contextes formels de type : services QdS et des contextes relationnels de type : services services pour chaque mode de composition. Les FTR obtenus permettent d'identifier les concepts contenant les services composites qui requièrent le minimum d'adaptation et qui offrent le maximum de QdS.

4.5 Coordination des services logiciels

Pour coordonner les services sélectionnés, nous proposons de générer automatiquement par transformation de modèles des processus d'orchestration BPEL à partir de

modèles intentionnels de services MIS [54].

4.5.1 Introduction aux transformations de modèles

Les transformations de modèles sont au coeur de l'approche de l'ingénierie dirigée par les modèles. Ci-dessous nous donnons quelques définitions reformulées par Hubert Kadima [50] :

Définition 1. *Une transformation de modèle est une opération qui consiste à générer un ou de plusieurs modèles cibles à partir d'un ou de plusieurs modèles sources conformément à une définition de transformation.*

Définition 2. *Une définition de transformation est un ensemble de règles de transformation qui décrivent globalement comment un modèle décrit dans un langage source peut être transformé en un modèle décrit dans un langage cible.*

Définition 3. *Une règle de transformation est une description de la manière dont un ou plusieurs éléments du modèle source peuvent être transformés en un ou plusieurs éléments du modèle cible.*

4.5.2 BPEL

Nous rappelons que BPEL ou BPEL4WS (*Business Process Execution Language for Web Services*) est un langage exécutable standardisé par l'OASIS et basé sur XML. Nous avons choisi d'utiliser BPEL parce qu'il s'agit du langage d'orchestration qui a gagné la bataille des standards en étant le langage le plus utilisé actuellement par l'industrie. BPEL repose sur un modèle constitué d'un workflow d'activités qui peuvent être de deux types : de base ou structurées. Les activités de base sont utilisées pour l'invocation d'une opération d'un service (`invoke`, `receive`, `reply`, `assign`, `wait`, `throw` et `terminate`). Elles manipulent un certain nombre de variables et elles sont liées à des `partnerLinks`. La conversation entre partenaires est assurée par des `correlationSets`. Les activités structurées sont basées sur l'approche des hiérarchies d'activités dans laquelle un processus est spécifié par le raffinement progressif d'une activité dans un arbre d'activités (`sequence`, `flow`, `while`, `switch`, `scope` et `pick`). Un modèle BPEL intègre aussi un mécanisme de gestion des exceptions `faultHandlers`, des erreurs `compensationHandlers` et des compensations `eventHandlers`.

4.5.3 Principe de la transformation

La transformation de modèles MIS vers BPEL se déroule en deux étapes successives. Dans une première étape, un premier modèle BPEL dit *intentionnel* est élaboré. Ce modèle est non exécutable et constitué de services intentionnels extraits des cartes et

spécifiés par les modèles MIS. Dans une deuxième étape, un second modèle BPEL dit *opérationnel* est fourni. Ce modèle est exécutable et constitué de services opérationnels obtenus suite aux étapes de découverte et de sélection de notre approche. La première étape de cette transformation est réalisée en considérant le méta-modèle de MIS et le méta-modèle BPEL 2.0. Un ensemble de règles ont été définies afin de mettre en œuvre cette transformation. Elles sont résumées dans le Tableau 4.3.

Élément MIS	**Équivalent BPEL**
Service atomique	`invoke`
Situation initiale	`receive`
Situation finale	`reply`
Pré-condition, Post-condition	`assign`
Lien de composition Séquence	`sequence`
Lien de composition Parallèle	`flow`
Lien de composition itératif	`while`
Point de variation Alternatif	`switch, if else`
Point de variation De chemin	`scope`
Point de variation Multiple	`pick`

Tableau 4.3 — Règles de transformation de MIS vers BPEL.

Un *Service atomique* est la représentation intentionnelle du service opérationnel, il est donc transformé vers l'activité de base `invoke` qui permet l'invocation d'un service. Les paramètres d'entrée *Situation initiale* sont introduits par `receive` et les paramètres de sortie *Situation finale* sont fournis par `reply`. Les *Pré-condition* et *Post-condition* sont transformées vers `assign`. Pour les services agrégats, le lien de composition *Séquence*, *Parallèle* et *itératif* sont transformés respectivement vers `sequence`, `flow` et `while`. Les points de variation *Alternatif*, *De chemin*, et *Multiple* sont transformés respectivement vers `switch` ou `if else`, `scope` et `pick`. Le résultat de cette première étape de transformation est une instance du méta-modèle BPEL 2.0 constitué de services intentionnels.
La première étape de la transformation est faite sur quatre passes :

1. La première passe consiste à créer les éléments du processus d'orchestration BPEL et ceci en considérant les règles de transformation définies entre MIS et BPEL.

2. La deuxième passe consiste à construire les activités structurées BPEL en identifiant les liens entre les différents éléments obtenus suite à la première passe.

3. La troisième passe consiste à identifier l'ordre d'exécution des activités de base et structurées BPEL.

4. La quatrième passe consiste à construire l'activité structurée représentant tout le processus d'orchestration BPEL.

La première étape fournit une instance du méta-modèle BPEL constituée par des services intentionnels. La deuxième étape de transformation permet alors de remplacer ces services intentionnels par les services logiciels sélectionnés et transformer l'instance BPEL en un processus exécutable.

4.6 Validation de la satisfaction des exigences

À cette étape nous proposons de valider la satisfaction des exigences par rapport aux applications à base de services construites. Pour ceci, nous utilisons deux approches : une approche empirique assurant la validation des exigences fonctionnelles [54] [28] et une approche automatique assurant la validation des exigences non-fonctionnelles [27].

4.6.1 Validation empirique des exigences fonctionnelles

Cette approche consiste à vérifier manuellement que chaque service logiciel composé satisfait au mieux les exigences fonctionnelles des utilisateurs. Pour ceci, deux mesures empruntées du domaine de recherche d'information sont calculées : la précision et le rappel [35]. Ces deux mesures ont été aussi très fréquemment utilisées dans le cadre des travaux qui relatent à la problématique de la découverte et la sélection des services Web [24] [73] [103].
Dans notre approche, la précision évalue le nombre de réels services pertinents et de haute QdS parmi les services sélectionnés par notre approche, tandis que le rappel évalue le nombre de services sélectionnés par notre approche parmi les réels services pertinents et de haute QdS, selon les équations suivantes :

$$\text{Précision} = \frac{\left|\left\{\text{Réels services pertinents et de haute QdS}\right\} \cap \left\{\text{Services sélectionnés par notre approche}\right\}\right|}{\left|\left\{\text{Services sélectionnés par notre approche}\right\}\right|}$$

$$\text{Rappel} = \frac{\left|\left\{\text{Réels services pertinents et de haute QdS}\right\} \cap \left\{\text{Services sélectionnés par notre approche}\right\}\right|}{\left|\left\{\text{Réels services pertinents et de haute QdS}\right\}\right|}$$

4.6.2 Validation automatique des exigences non-fonctionnelles

Cette approche consiste à vérifier automatiquement que chaque service logiciel composé satisfait au mieux les exigences non-fonctionnelles. La technique utilisée est la simulation. La simulation est une technique qui permet l'imitation du fonctionnement d'un système réel à travers un modèle. Une évaluation par simulation est le résultat d'analyse de plusieurs scénarii de ce modèle. Nous avons opté pour la simulation parce

qu'elle présente un moyen d'évaluation flexible permettant de considérer différentes variantes du contexte d'exécution. De ce fait, elle s'avère appropriée à l'évaluation des applications à base de services Web qui évoluent dans des contextes d'exécution très dynamiques. De plus, cette technique a prouvé sa riche expressivité et son efficacité en terme d'analyse de performances dans divers domaines tels que : les réseaux de communication, les applications à temps réel, etc. Il existe quatre types de simulation [23] : la simulation continue, la simulation à évènements discrets, la simulation conduite par trace et l'émulation.

- La simulation continue repose sur des modèles contenant en grande partie des variables continues qui évoluent dans le temps. La simulation de ces modèles revient à résoudre des systèmes d'équations différentielles.

- La simulation à évènements discrets utilise un modèle discret du système. Un évènement possède un instant d'exécution indiquant quand il doit se produire. Les variables du modèle changent à des instants de déclenchement des évènements. Le temps est représenté par une variable appelée horloge de simulation.

- La simulation conduite par trace est utilisé si le système existe et s'il peut être observé. Pour ce type de simulation, les entrées du modèle à simuler peuvent provenir d'échantillons obtenus directement du système par des techniques de mesure au lieu d'employer des valeurs aléatoires.

- L'émulation est une cohabitation entre la simulation et une partie du système réel.

La simulation à évènements discrets s'avère la technique la plus appropriée. D'abord, vu qu'elle cible la résolution des systèmes d'équations différentielles, la simulation continue est donc un choix à écarter. Ensuite, la technique de simulation conduite par trace et l'émulation sont inappropriées puisqu'elles se lient étroitement au système et à l'environnement dans lequel il se situe, tout au plus elles procurent un coût considérable. Le modèle utilisé est illustré par la Figure 4.9. Ce modèle est constitué de deux parties [25] [27] :

- **Le modèle de l'application distribuée** : le fonctionnement de l'application distribuée à base de services Web suit le modèle client-serveur. Le rôle du client consiste à envoyer des requêtes d'invocation au service. Le serveur qui héberge le service invoqué transmet ces requêtes d'invocation et renvoie les messages réponses au client. La communication entre le client et le service est réalisée moyennant différents types d'actions. Le modèle de l'application distribuée est obtenu en définissant des correspondances entre ces actions et les activités de base et structurées du processus d'orchestration BPEL. En formalisme évènements discrets, les activités de base BPEL sont modélisés par une suite

d'actions de `traitement`, d'`invocation`, de `lecture` et d'`écriture`. Les activités structurées BPEL sont modélisées par une suite d'actions de `transfert` et de `synchronisation`. Ci-dessous, nous expliquons ces actions :

- `Traitement` : cette action est faite par le service et elle consiste à exécuter l'opération invoquée en vue de produire le résultat attendu par le client.
- `Invocation` : cette action est faite par le client et elle consiste à interroger un service en vue d'exécuter l'une de ces opérations. Cette action prend trois paramètres : le nom du service, l'opération à invoquer et les paramètres d'entrée de cette opération.
- `Lecture` : cette action est faite par le serveur lors de la réception d'une demande d'invocation de la part d'un client. Elle permet de lire les paramètres d'entrée nécessaires à l'exécution d'une opération d'un service.
- `Ecriture` : cette action est faite par le serveur lors de la réception d'une demande d'invocation de la part d'un client. Elle permet de stocker les paramètres d'entrée nécessaires à l'exécution d'une opération d'un service et ceci en vue d'une réutilisation ultérieure.
- `Transfert` : cette action est faite par le protocole applicatif SOAP assurant la communication et le transfert de données entre le client et le serveur hébergeant le service invoqué.
- `Synchronisation` : cette action est faite par le protocole SOAP et elle consiste à synchroniser les différentes actions faites par le client ou le serveur.

– **Le modèle de l'infrastructure du réseau** : l'infrastructure réseau est modélisée par un réseau local et par un réseau internet. Ces réseaux sont composés par des nœuds de traitement et d'autres de relais. Les données échangées sont véhiculées par un canal de communication.

Les tests de simulation réalisés permettent de calculer des métriques de QdS par analyse des traces de simulation récupérées. Nous proposons d'évaluer deux propriétés de QdS qui sont le temps de réponse et la disponibilité selon les métriques suivantes [26] :

- Le temps de réponse T d'un service Web s étant défini ainsi : T (s) = S(s) + M(s) avec :
  - S(s) (Temps de Service) : temps que prend le service pour exécuter l'opération invoquée.
  - M(s) (Temps d'´echanges de Messages) : temps nécessaire pour envoyer et

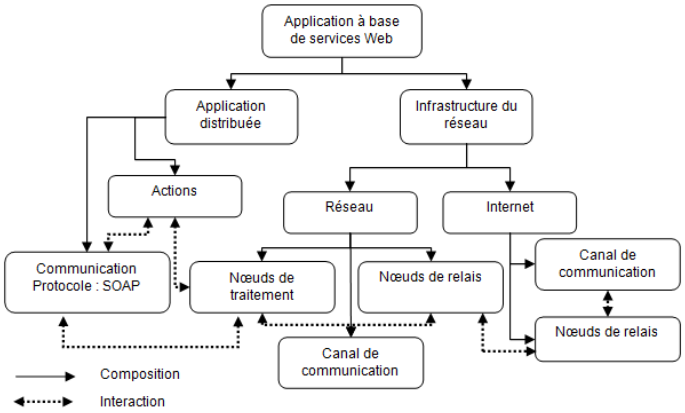


Figure 4.9 — Modèle de simulation à évènements discrets des applications à base de services Web.

recevoir les messages SOAP.

– La disponibilité D d'un service Web s est calculée ainsi : D(s)= 100*(1-[Nombre d'invocations rejetées(I)/ Nombre d'invocations accomplies(I)]) et ce pendant un intervalle de temps I.

Ces deux métriques permettent de fournir des mesures locales que nous utilisant pour déduire les mesures globales de toute la composition et ceci en appliquant le modèle mathématique de Cardoso *et al.* [13]. Ce modèle consiste à calculer une mesure globale relative à chacun des blocs d'interaction de la composition : bloc séquentiel, parallèle, conditionnel et itératif. La mesure globale sera donc la somme des mesures de ces blocs. Le Tableau 4.4 illustre ce modèle mathématique.

4.7 Conclusion

Dans le cadre de cette thèse, nous avons proposée une nouvelle approche multi-perspective centrée exigences pour la composition de services Web. Cette approche assure l'alignement des services Web aux exigences des utilisateurs tout en considérant différentes facettes de la variabilité essentielle et technique. Les contributions apportées dans le cadre de ce travail sont :

– La définition et la caractérisation de deux perspectives pour la composition de services Web : une perspective centrée exigences appelée perspective "intentionnelle" qui intègre la variabilité essentielle au niveau modèle de spécification des services métiers et une perspective centrée fonctions appelée perspective "opérationnelle" qui intègre la variabilité technique au niveau de la coordination

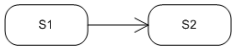
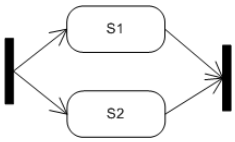
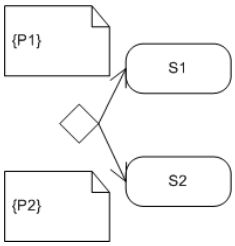
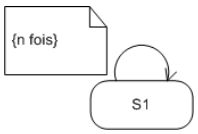
Bloc d'interaction	Schéma	Mesure globale
Séquentiel		T=T(s_1) + T(s_2) D=D(s_1)×D(s_2)
Parallèle		T=max(T(s_1), T(s_2)) D=D(s_1)×D(s_2)
Conditionnel		T=p_1×T(s_1) + p_2×T(s_2) D=p_1×D(s_1) + p_2×D(s_2)
Itératif		T=$\sum_{i=1}^{n} T(s_i)$ $D = \prod_{i=1}^{n} D(s_i)$

Tableau 4.4 — Modèle mathématique de Cardoso *et al.* [13] de calcul de mesures globales de QdS à partir des mesures locales

et de l'exécution des services logiciels.

– L'extension du modèle intentionnel de services par une couche de qualité de service QdS.

– La proposition d'une méthode de découverte automatique des services logiciels pertinents. La découverte est réalisée en interrogeant le moteur de recherche avec un ou plusieurs mots-clés extraits à partir des spécifications des services métiers. Pour réduire l'ensemble des services logiciels retourné par le moteur de recherche et pour faciliter la sélection par la suite, différents niveaux de filtrage (de QdS, syntaxique et sémantique) sont appliqués pour en garder que les services pertinents qui répondent le mieux aux exigences fonctionnelles.

– La définition et la mise en œuvre d'une méthode de guidage automatique pour la sélection des services logiciels pertinents et de haute QoS. Cette méthode est basée sur deux cadres de travail formels qui sont l'Analyse Formelle de Concepts AFC et l'Analyse Relationnelle de Concepts ARC et elle s'applique pour la

sélection des services logiciels atomiques et la sélection des services logiciels composites.

– L'implémentation d'un mécanisme de génération automatique des procédés de coordination des services logiciels à partir des spécifications des services métiers se basant sur la technique de transformation de modèles.

– La validation de la composition : empiriquement en termes de précision et de rappel et automatiquement en utilisant la technique de simulation.

Dans le chapitre qui suit, nous présentons des validations expérimentales de l'approche proposée.

Chapitre 5

Expérimentations

Dans le chapitre précédent, nous avons présenté notre approche proposée pour la composition centrée exigences des services Web. Nous nous intéressons dans ce chapitre à illustrer et à expérimenter cette approche sur un cas d'étude d'une application à base de services d'arrangement de conférences. Cette application permet l'arrangement du transport, du logement et du divertissement aux participants d'une conférence. Nous validons empiriquement notre approche en termes de rappel et de précision sur cette application. Nous utilisons également la simulation pour en déduire la politique de composition qui satisfait le mieux les exigences des utilisateurs.

Sommaire

5.1 Modélisation intentionnelle des services métiers

La modélisation intentionnelle des services métiers est faite en utilisant le formalisme la Carte. La Figure 5.1 représente les exigences des utilisateurs pour notre application d'arrangement de conférences.

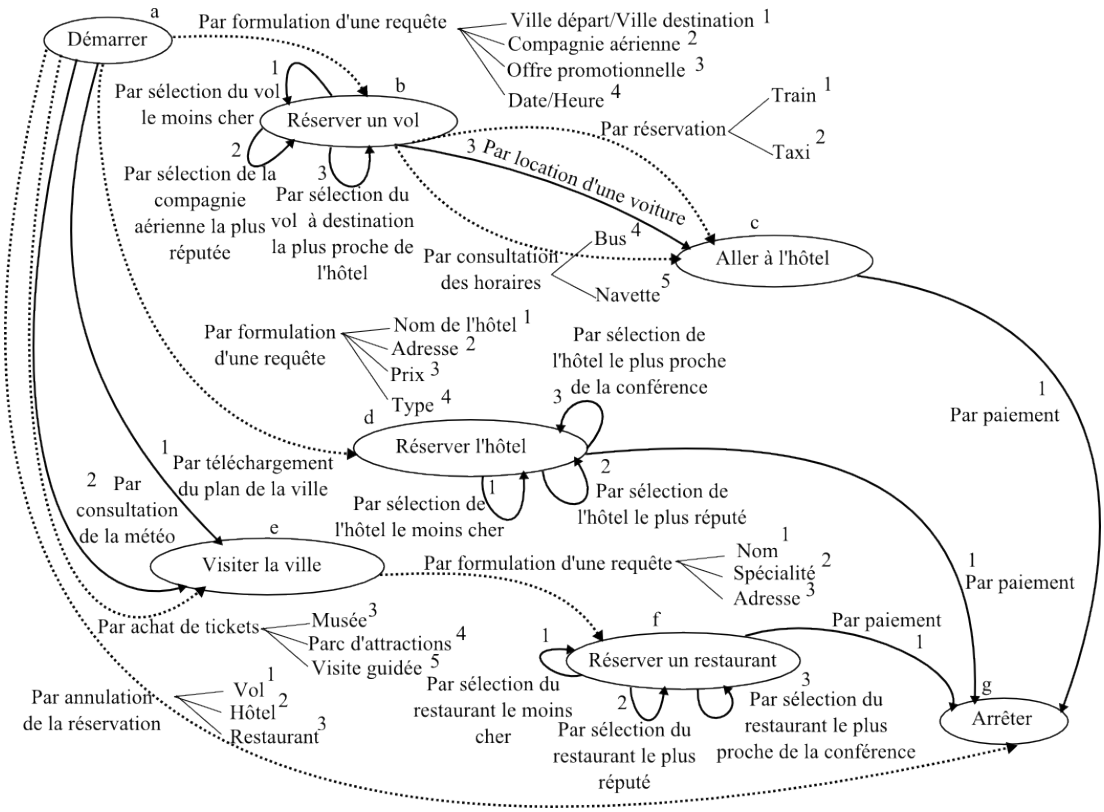


Figure 5.1 — La carte d'une application à base de services d'arrangement de conférences.

La carte de cette application a cinq intentions : *Réserver un vol*, *Aller à l'hôtel*, *Réserver un hôtel*, *Visiter la ville* et *Réserver un restaurant*, et a plusieurs stratégies permettant de satisfaire chacune de ces intentions. Par exemple, pour satisfaire l'intention Visiter la ville, les utilisateurs peuvent suivre deux stratégies différentes : *Par téléchargement du plan de la ville* ou *Par achat de tickets*.

Nous trouvons dans cette carte différent types de relations entre les sections :

– <*Démarrer, Réserver un vol, Par formulation d'une requête*> est un paquet composé de quatre sections ayant chacune une stratégie différente : *Par ville départ/ville arrivée*, *Par compagnie aérienne*, *Par offre promotionnelle*, et enfin

Par date. Ce paquet est noté $P_{ab} = \otimes(ab_1, ab_2, ab_3, ab_4)$ [1].

- les trois stratégies *Par sélection de l'hôtel le moins cher*, *Par sélection de l'hôtel le plus réputé* et *Par sélection de l'hôtel le plus proche de la conférence* représentent trois façons différentes pour *Réserver un hôtel*. Les trois sections <*Réserver un hôtel, Réserver un hôtel, Par sélection de l'hôtel le moins cher*>, <*Réserver un hôtel, Réserver un hôtel, Par sélection de l'hôtel le plus réputé*> et <*Réserver un hôtel, Réserver un hôtel, Par sélection de l'hôtel le plus proche de la conférence*> forment un multi-segment. Ce multi-segment est noté $MSdd = \vee(dd1, dd2, dd3)$.

- les trois sections <*Démarrer, Réserver un vol, Par formulation d'une requête : ville départ/ville arrivée*>, <*Réserver un vol, Réserver un vol, Par sélection du vol le moins cher*> et <*Réserver un vol, Aller à l'hôtel, Par location d'une voiture*> constituent un chemin. Ce chemin est noté $C_{a,b,c} = \bullet(ab_1, bb_1, bc_3)$.

- il y a trois chemins distincts pour atteindre l'intention *Arrêter* depuis l'intention *Démarrer*. Le premier est le chemin via les intentions *Réserver un vol* et *Aller à l'hôtel*. Le deuxième est le chemin via l'intention *Réserver un hôtel*. Et le troisième est le chemin via les intentions Visiter la ville et *Réserver un restaurant*. Ce multi-chemin est noté $MC_{a,b,c,d,e,f,g} = \cup(C_{a,b,c,g}, C_{a,d,g}, C_{a,e,f,g})$.

La Figure 5.2 montre l'affinement de la section <*Réserver un hôtel, Arrêter, Par paiement*> au niveau i par une carte au niveau i+1. Cette carte propose plusieurs chemins entre *Démarrer* et *Arrêter*. Chacun de ces chemins est équivalent à la section <*Réserver un hôtel, Arrêter, Par paiement*> du niveau i.

En appliquant les directives d'extraction de services intentionnels à partir de deux cartes précédentes, nous identifions 39 services atomiques. Ces services sont décrits par le Tableau 5.1.

5.2 Découverte des services logiciels

Dans ce qui suit, nous allons montrer comment découvrir les services logiciels correspondant aux services intentionnels de notre application d'arrangement de conférences. Nous considérons le service $S_{Convertir\ la\ devise}$ présenté par la section cc1 de la carte de niveau i+1. Ce service assure la conversion de devises. La Figure 5.3 montre le modèle intentionnel du service $S_{Convertir\ la\ devise}$ qui est un fichier .xmi. Ce

1. Nous utilisons une codification locale pour désigner une section d'une carte. Les intentions sont codées par des lettres de l'alphabet. Les stratégies sont numérotées relativement à leurs intentions source et cible. Les niveaux d'affinement d'une carte vont de i à i+n. Par exemple, la section ab1i est une section de la carte de niveau i, son intention source est a, son intention cible est b et la stratégie permettant de réaliser l'intention b à partir de a est la stratégie 1.

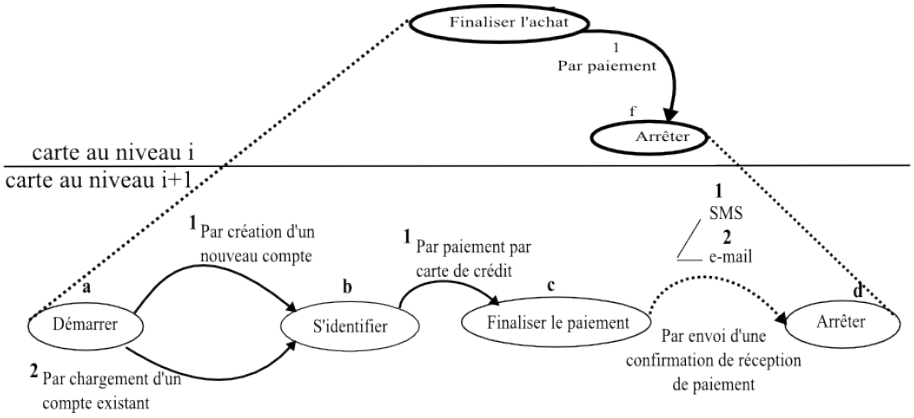


Figure 5.2 — L'affinement de la section <Réserver un hôtel, Arrêter, Par paiement>.

fichier est obtenu en instanciant le méta-modèle MIS en utilisant l'outil Kermeta [2].

L'interrogation du moteur de recherche de services Web *Service-Finder* se fait par mots-clés et nous utilisons pour notre exemple les mots clés : "Convertir + Devise". Ces mots-clés sont extraits à partir de la spécification intentionnelle du service métier $S_{Convertir\ la\ devise}$ présenté par la Figure 5.3. L'extraction des mots-clés est faite en appliquant la méthode TF-IDF sur le fichier .xmi instancié du méta-modèle de MIS présenté dans la Figure 5.3 et plus précisément sur les valeurs des champs *intention id, intention description, pre_condition valeur, post_condition valeur, ressource.0 nom, ressource.1 nom, et ressource.2 nom.* Les deux mots ayant les pondérations les plus élevées sont "Convertir" et "Devise" et par conséquence sont les mots-clés de la spécification intentionnelle du service métier $S_{Convertir\ la\ devise}$. En réponse à cette requête, *Service-Finder* retourne un ensemble de 76 services Web [3] comme le montre la Figure 5.4.

Pour réduire cet ensemble de services, nous procédons à un filtrage à trois niveaux. Au premier niveau, nous éliminons les services redondants. Pour le service $S_{Convertir\ la\ devise}$ et après ce premier niveau de filtrage, nous avons un ensemble de 46 services parmi les 76 retournés par *Service-Finder*. Au deuxième niveau, nous examinons les services qui restent selon deux propriétés de QdS à savoir la validité et la disponibilité. En appliquant ce deuxième niveau de filtrage, nous obtenons un nouvel ensemble formé de 26 services seulement. Ces services sont passés à un troisième niveau de filtrage qui est basé sur une mesure de similarité sémantique entre les modèles MIS des services intentionnels et les spécifications WSDL des services opérationnels. Pour ceci, nous avons utilisé une fonctionnalité qui compare la similarité de deux fichiers .xml implémentée et disponible au niveau de la librairie JWSL (*Java WordNet Similarity Library*). Avec cette fonctionnalité, un seuil empirique de 0.8 permet de discriminiser la similarité ou la dissimilarité sémantique entre deux fichiers xml. Ainsi, seuls les ser-

2. http ://www.kermeta.org/
3. Ce résultat est obtenu le 20 janvier 2011.

Section	Service métier
ab1i	$S_{Formuler\ une\ requête\ :\ ville\ départ/ville\ d'arrivée}$
ab2i	$S_{Formuler\ une\ requête\ :\ compagnie\ aérienne}$
ab3i	$S_{Formuler\ une\ requête\ :\ offre\ promotionnelle\ de\ vols}$
ab4i	$SS_{Formuler\ une\ requête\ :\ date\ du\ vol}$
bb1i	$S_{Sélectionner\ le\ vol\ le\ moins\ cher}$
bb2i	$S_{Sélectionner\ la\ compagnie\ aérienne\ la\ plus\ réputée}$
bb3i	$S_{Sélectionner\ la\ destination\ la\ plus\ proche\ de\ l'hôtel}$
bc1i	$S_{Réserver\ un\ train}$
bc2i	$S_{Réserver\ un\ taxi}$
bc3i	$S_{Louer\ une\ voiture}$
bc4i	$S_{Consulter\ les\ horaires\ des\ bus}$
bc5i	$S_{Consulter\ les\ horaires\ de\ la\ navette}$
ad1i	$S_{Formuler\ une\ requête\ :\ nom\ de\ l'hôtel}$
ad2i	$S_{Formuler\ une\ requête\ :\ adresse\ de\ l'hôtel}$
ad3i	$S_{Formuler\ une\ requête\ :\ prix\ de\ l'hôtel}$
ad4i	$S_{Formuler\ une\ requête\ :type\ de\ l'hôtel}$
dd1i	$S_{Sélectionner\ l'hôtel\ le\ moins\ cher}$
dd2i	$S_{Sélectionner\ l'hôtel\ le\ plus\ réputé}$
dd3i	$S_{Sélectionner\ l'hôtel\ le\ plus\ proche}$
ae1i	$S_{Télécharger\ le\ plan\ de\ la\ ville}$
ae2i	$S_{Consulter\ la\ météo}$
ae3i	$S_{Acheter\ un\ ticket\ de\ musée}$
ae4i	$S_{Acheter\ un\ ticket\ de\ parc\ d'attractions}$
ae5i	$S_{Acheter\ un\ ticket\ de\ visite\ guidée}$
ef1i	$S_{Formuler\ une\ requête\ :\ nom\ du\ restaurant}$
ef2i	$S_{Formuler\ une\ requête\ :\ spécialité\ du\ restaurant}$
ef3i	$S_{Formuler\ une\ requête\ :\ adresse\ du\ restaurant}$
ff1i	$S_{Sélectionner\ le\ restaurant\ le\ moins\ cher}$
ff2i	$S_{Sélectionner\ le\ restaurant\ le\ plus\ réputé}$
ff3i	$S_{Sélectionner\ le\ restaurant\ le\ plus\ proche}$
ag1i	$S_{Annuler\ la\ réservation\ d'un\ vol}$
ag2i	$S_{Annuler\ la\ réservation\ d'un\ hôtel}$
ag3i	$S_{Annuler\ la\ réservation\ d'un\ restaurant}$
ab1i+1	$S_{Créer\ un\ compte}$
ab2i+1	$S_{Charger\ un\ compte}$
bc1i+1	$S_{Payer\ avec\ carte\ de\ crédit}$
cc1i+1	$S_{Convertir\ la\ devise}$
cd1i+1	$S_{Envoyer\ un\ sms}$
cd2i+1	$S_{Envoyer\ un\ e\text{-}mail}$

Tableau 5.1 — Services métiers atomiques de l'application d'arrangement de conférences représentée par la série de cartes

vices logiciels dont les spécifications WSDL ont une valeur de similarité sémantique normalisée supérieure ou égale à 0.8 avec les modèles MIS des services intentionnels auxquels ils correspondent sont maintenus. Ce troisième niveau de filtrage génère un nouvel ensemble qui renferme 9 services Web seulement. Ces 9 services sont les services pertinents qui répondent le mieux à l'intention *Convertir la devise*. Dans le Tableau 5.2,

```
<?xml version="1.0" encoding="ASCII"?>
<mis:carte xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:mis="http://ism/1.0"
xsi:schemaLocation="http://mis/1.0 MIS.ecore">
 <service xsi:type="ism:Atomique"
 intention="//@Intention.0"
 pre_condition="//@Pre-Condition.0"
 post_condition="//@Post-Condition.0"
 situation_initiale="//@Situation Initiale.0"
 situation_finale="//@Situation Finale.0"
 ressource.0="//@Ressource.0"
 ressource.1="//@Ressource.1"
 ressource.2="//@Ressource.2"
 propriete_qds.0="//@Propriete QdS.0"
 propriete_qds.1="//@Propriete QdS.1"
 priorite.0="//@Priorite.0"
 priorite.1="//@Priorite.1"
 preference.0="//@Preference.0"
 preference.0="//@Preference.0"
 id="Convertir la devise"/>
 <Intention description="Convertir la devise"/>
 <pre_condition valeur="Reservation.Etat=vrai"/>
 <post_condition valeur="Conversion.Valeur<>0"/>
 <ressource.0 nom="Devise"/>
 <ressource.1 nom="Reservation"/>
 <ressource.2 nom="Conversion"/>
 <situation_initiale input="//@resource.0 //@resource.1"/>
 <situation_finale output="//@resource.2"/>
 <propriete_qds.0 id="temps de reponse">
  <priorite.0 valeur="3"/>
  <preference.0 valeur="Tres faible"/>
 </propriete_qds.0>
 <propriete_qds.1 id="Disponibilite">
  <priorite.1 valeur="1"/>
  <Preference.1 valeur="Tres eleve"/>
 </Propriete_qds.1>
</mis:carte>
```

Figure 5.3 — La spécification intentionnelle du service métier $S_{Convertir\ la\ devise}$.

nous donnons un résumé du nombre de services logiciels obtenus après interrogation de *Service-Finder* et après chaque niveau de filtrage.

	Service-Finder	**Filtrage**		
Nombre de services	76	1^{er} **niveau**	2^{e} **niveau**	3^{e} **niveau**
		46	26	9

Tableau 5.2 — Résultat de l'étape de découverte du service intentionnel $S_{Convertir\ la\ devise}$

5.3 Sélection des services logiciels pertinents et de haute QdS

5.3.1 Sélection des services logiciels atomiques

Pour notre problème de sélection de services atomiques, nous définissons un contexte K où les individus sont les services pertinents obtenus après l'étape de

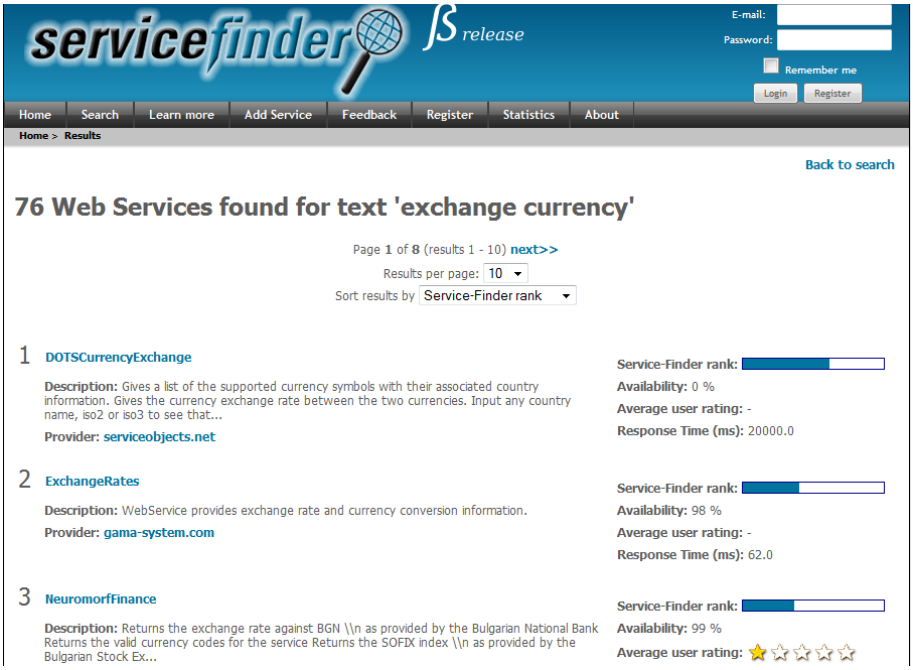


Figure 5.4 — Résultat d'interrogation de *Service-Finder* à la requête "Convertir + Devise".

découverte et les propriétés représentent des propriétés de QdS.

Comme spécifié dans le modèle intentionnel du service $S_{Convertir\ la\ devise}$ présenté par la Figure 5.3, les utilisateurs donnent trois fois plus d'importance au temps de réponse qu'à la disponibilité (la priorité accordée au temps de réponse est 3, tandis que la priorité accordée à la disponibilité est 1). Pour tenir compte de cette priorité, nous définissons pour chaque valeur ordinale du temps de réponse trois sous-valeurs. Par exemple, nous considérons trois sous-valeurs : très faible 1, très faible 2 et très faible 3 pour la valeur ordinale très faible. Un service ayant un temps de réponse très faible devrait avoir une relation d'incidence avec les trois sous-valeurs de la valeur ordinale très faible. Aussi, nous considérons que si un service a une relation d'incidence avec une valeur ordinale du temps de réponse (avec une valeur ordinale de la disponibilité, respectivement), alors il devrait avoir une relation d'incidence avec toutes les valeurs ordinales qui viennent juste après (qui viennent juste avant, respectivement). Par exemple, si un service a un temps de réponse très faible (une disponibilité très élevée, respectivement), alors il a également un temps de réponse faible, moyen, élevé et très élevé (une disponibilité élevée, moyenne, faible et très faible, respectivement). Les services pertinents et de haute QdS sont alors les services qui ont le plus de relations d'incidence avec les propriétés du contexte.

Le Tableau 5.3.1 donne une vue partielle du contexte K. Il montre les 9 services pertinents obtenus après l'étape de découverte (les individus en lignes), et leur(s) rela-

tion(s) “est caractérisé par” avec la disponibilité et le temps de réponse (les propriétés en colonnes).

	(D1) Disponibilité très faible	(D2) Disponibilité faible	(D3) Disponibilité moyenne	(D4) Disponibilité élevée	(D5) Disponibilité très élevée	(TR11) Temps de réponse très faible 1	(TR12) Temps de réponse très faible 2	(TR13) Temps de réponse très faible 3	...	(TR51) Temps de réponse très élevé 1	(TR52) Temps de réponse très élevé 2	(TR53) Temps de réponse très élevé 3
ExchangeRates	×	×	×	×		×	×	×	...	×	×	×
CurrencyServer	×	×	×	×	×	×	×	×	...	×	×	×
CurrencyService	×	×	×	×	×	×	×	×	...	×	×	×
ForeignExchangeRates	×	×	×	×	×	×	×	×	...	×	×	×
ExchangeRateWebService	×	×	×						...	×	×	×
ForeignExchangeService	×	×	×	×					...	×	×	×
CurrencyRequest	×	×	×	×					...	×	×	×
Financial	×	×	×			×	×	×	...	×	×	×
IOSXMLReq	×	×	×	×					...	×	×	×

Tableau 5.3 — Contexte K reliant les services aux propriétés de QdS.

La Figure 5.5 représente le treillis de concepts associé à notre contexte K. Ce treillis est construit en utilisant l'outil Galicia [4](*Galois Lattice Interactive Constructor*). Galicia est une plate-forme open-source permettant la construction, la visualisation et le stockage des treillis de concepts. Le treillis présenté par la Figure 5.5 est avec un étiquetage réduit : un concept de ce treillis est défini par l'ensemble des individus des concepts situés en-dessous et des propriétés des concepts situés au-dessus. L'exploration du treillis nous a permis de conclure que les services pertinents qui offrent le meilleur compromis entre disponibilité et temps de réponse sont les services de l'extension du concept c1 se trouvant en bas du treillis. L'extension de ce concept contient 3 services : `CurrencyServer`, `CurrencyService` et `ForeignExchangeRates`. Ces services ont une disponibilité très élevée (D5) et un temps de réponse très faible (TR11, TR12 et TR13). Ces trois services sont interchangeables : en cas de panne ou d'indisponibilité de l'un de ces services, ce dernier peut être remplacé dynamiquement par l'un des deux services qui restent et ceci tout en préservant la même QdS demandée.

5.3.2 Sélection des services logiciels composites

Considérons le service intentionnel $S_{Consulter\ la\ météo}$. Ce service peut être vu comme un service composite constitué de trois opérations Op1, Op2 et Op3 s'exécutant séquentiellement : Op1 permet de retourner le nom de la ville à partir d'une adresse IP. Op2 fournit le code zip d'une ville. Et Op3 renvoie, à partir d'un code zip, les informations météo correspondantes. Pour découvrir les services correspondants aux opérations Op1, Op2 et Op3, nous proposons d'interroger le moteur de recherche *Service-Finder*

4. http ://galicia.sourceforge.net/

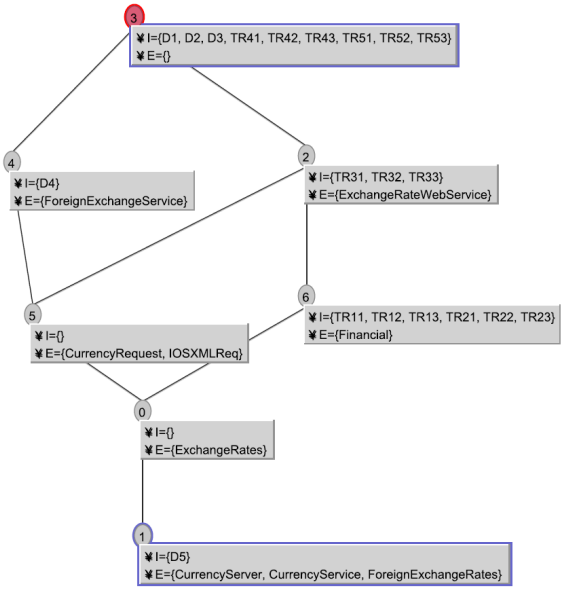


Figure 5.5 — Le treillis du contexte K.

en utilisant un ensemble de mots-clés et d'appliquer un filtrage à deux niveaux : de QdS et syntaxique et ceci afin de réduire l'ensemble des services retourné. Le Tableau 5.4 résume cette étape de découverte.

	Mots-clés	Nombre de services		Ensemble ID
		Service-Finder	Filtrage	
Op1	{IP, adresse, nom, ville}	94	17	S1.i
Op2	{nom, ville, zip, code, postal}	768	96	S2.j
Op3	{zip, code, postal, météo}	39	21	S3.k

Tableau 5.4 — Résultats de découverte des services logiciels correspondants aux opérations Op1, Op2 et Op3 du service intentionnel $S_{Consulter\ la\ météo}$.

Les mots-clés utilisés pour chercher les services logiciels correspondants à Op1 sont : "IP + adresse + nom + ville". L'interrogation de *Service-Finder* par ces mots-clés donne un ensemble formé de 94 services logiciels. Ces services sont filtrés selon deux critères. Le premier critère est la QdS ; il s'agit d'éliminer les services invalides et indisponibles. Le deuxième critère est syntaxique ; il s'agit de ne garder que les services compatibles. Le filtrage fournit alors un ensemble formé uniquement par 17 services logiciels. Nous notons cet ensemble S1.i. La même démarche s'applique pour chercher les services logiciels correspondants à Op2 et Op3. Nous obtenons alors un ensemble S2.j formé par 96 services pour Op2 et un ensemble S3.k formé par 21 services pour Op3.

Après cette étape de découverte, nous procédons à une classification des services selon leur mode de composition. Le Tableau 5.5 fournit les résultats de classification

des services de S1.i, S2.j et S3.k. Par exemple, pour les services de l'ensemble S2.j, nous avons uniquement 3 services qui sont entièrement composables avec les services S1.i, 89 services partiellement composables, 1 seul service adaptable-entièrement composable et 3 services adaptables-partiellement composables.

Ensemble de Services	Nombre de services			
	Entièrement composable (EC)	Partiellement composable (PC)	Adaptable-Entièrement composable (A-EC)	Adaptable-Partiellement composable (A-PC)
S1.i	12	4	2	0
S2.j	3	89	1	3
S3.k	12	4	11	1

Tableau 5.5 — Classification des services de S1.i, S2.j et S3.k selon leur mode de composition.

Pour pouvoir sélectionner les services composites qui requièrent le minimum d'adaptation et qui offrent le maximum de QdS, nous considérons une famille de contextes. Cette famille renferme trois contextes formels : S1.i × QdS, S2.j ×QdS, S3.k × QdS et huit contextes relationnels, quatre pour chaque paire de services consécutifs, de types : S1.i × S2.j et S2.j × S3.k décrivant chacun un mode de composition. Pour faciliter l'exploration des treillis obtenus par application de l'ARC sur cette famille de contextes, nous proposons d'intégrer un mécanisme de requêtes permettant de spécifier la QdS et le mode de composition désirés. Ce ci est fait en ajoutant une ligne requête au niveau de chaque contexte formel et relationnel.

Les services que nous désirons sélectionner pour cet exemple sont : les services entièrement composables et ayant un temps de réponse et une disponibilité élevés. Ces services figurent dans les concepts et les sous-concepts dont l'intention renferme cette requête. Les Figures 5.6, 5.7 et 5.8 présentent les treillis obtenus après application de l'ARC en utilisant l'outil Galicia.

Le treillis de la Figure 5.6 est celui de l'ensemble S1.i. Les services à sélectionner de S1.i sont les services du sous-concept c0 du concept c15 où figure la Requête1. Ces services sont S1.3, S1.5 et S1.59. Ces services sont entièrement composables (c18) et ont un temps de réponse et une disponibilité élevée (c11).

La Figure 5.7 présente le treillis de l'ensemble S2.j. Les services entièrement composables et ayant un temps de réponse et une disponibilité élevés sont les services S2.198 de c32 et S2.8 de c11. Ces services répondent à la Requête 2 qui figure dans l'extension du concept c31 et qui hérite la propriété temps de réponse élevé du concept c56 et la propriété disponibilité élevée du concept c43.

Finalement, la Figure 5.8 présente le treillis de l'ensemble des services S3.k. Les services à sélectionner sont les services qui répondent à la requête Requête3 de c50 et qui se trouvent dans les sous-concepts c0 et c35 : S3.1 et S3.2. Ces services sont entièrement

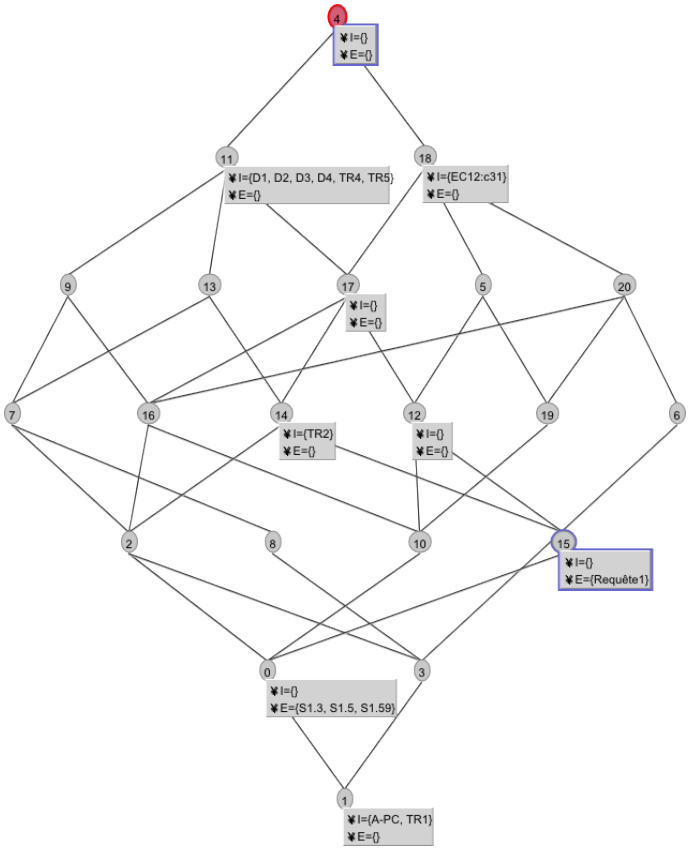


Figure 5.6 — Le treillis de S1.i présentant le résultat de la requête "Requête1" : services entièrement composables et ayant un temps de réponse et une disponibilité élevés.

composables et sont caractérisés par un temps de réponse et une disponibilité élevés (c24 et c40).

Les informations concernant les services sélectionnés sont données dans le Tableau 5.6. Par exemple, si nous voulons minimiser le temps de réponse, nous pouvons composer les services S1.59, S2.198 et S3.2 3 qui ont les temps de réponse les moins élevés.

5.4 Coordination des services logiciels

Le résultat de la première étape de transformation est une instance du méta-modèle BPEL 2.0 constitué de services intentionnels. Cette instance est un fichier .xmi incompréhensible par les moteurs d'orchestration. Pour pouvoir exécuter ce fichier, nous utilisons l'API *BPELWriter* disponible sous le plugin Eclipse BPEL Designer [5]. Cet API permet de générer des fichiers exécutables .bpel à partir de modèles BPEL. Pour

5. http ://www.eclipse.org/bpel/downloads.php

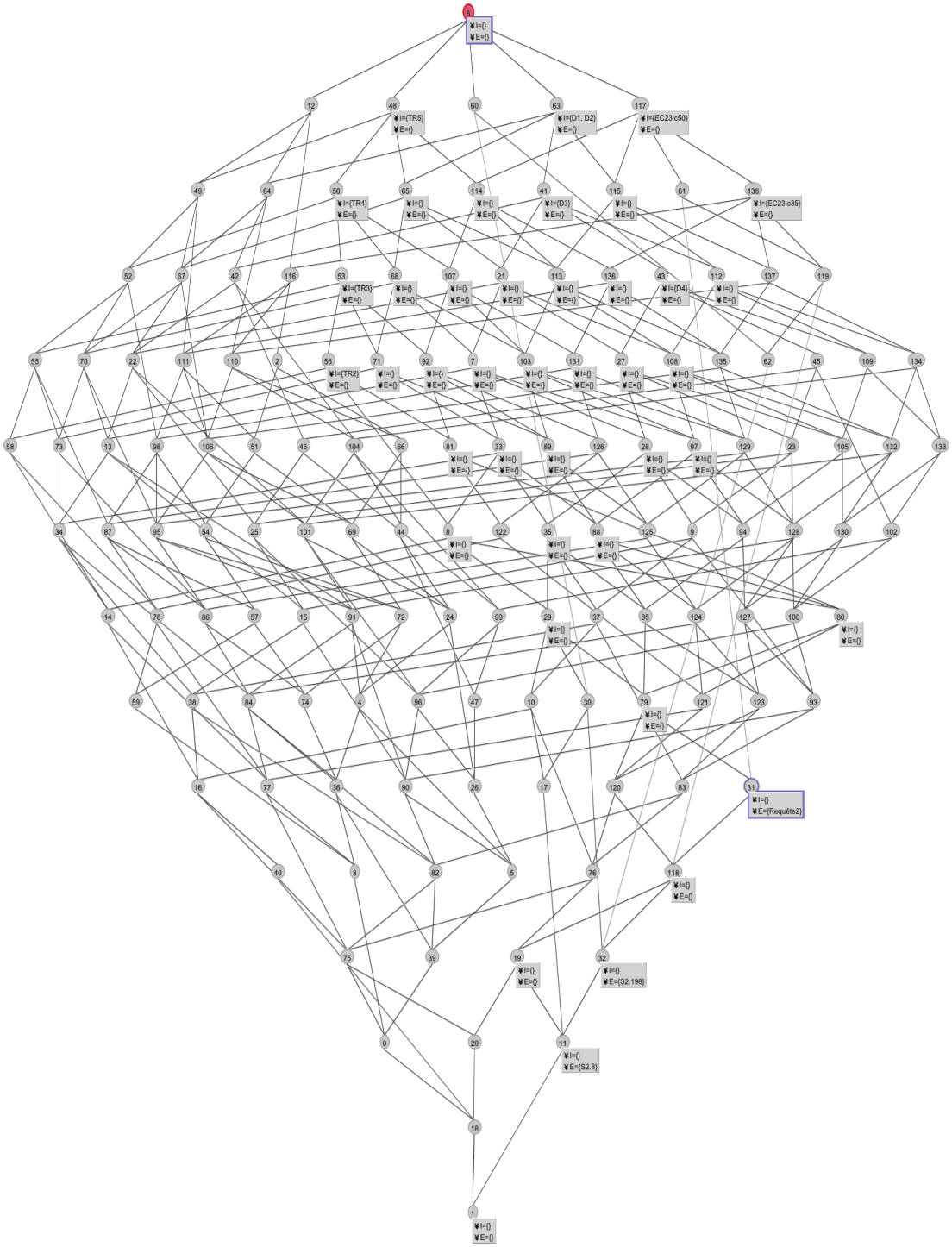


Figure 5.7 — Le treillis de S2.j présentant le résultat de la requête “Requête2” : services entièrement composables et ayant un temps de réponse et une disponibilité élevés.

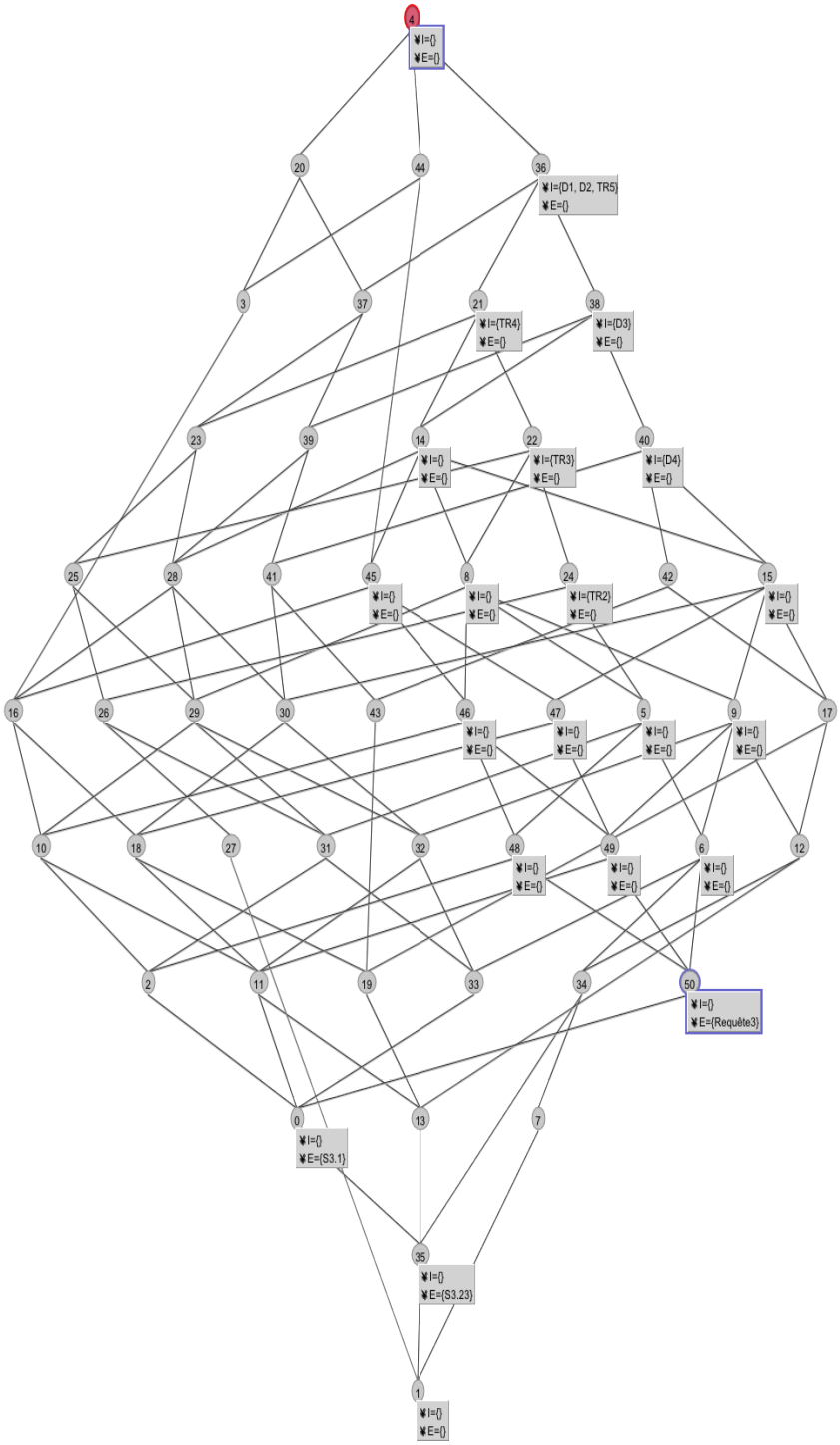


Figure 5.8 — Le treillis de S3.k présentant le résultat de la requête “Requête3” : services entièrement composables et ayant un temps de réponse et une disponibilité élevés.

Service	Nom	Opération	Disponibilité (%)	Temps de réponse(ms)

S1.59	`Ip2LocationWebService`	`IP2Location`
		`(in)IP: string`
		`(out)CITY: string`
			100	257

S1.5	`GeoCoder`	`IPAddressLookup`
		`(in)ipAdress: string`
		`(out)City: string`
			100	328

S1. 3	`IPGeo`	`ResolveIP`
		`(in)IpAddress: string`
		`(out)CITY: string`
			100	798

S2.198	`MediCareSupplier`	`GetSupplierByCity`
		`(in)City: string`
		`(out)ZIP: string`
			85	304

S2.8	`ZipcodeLookupService`	`CityToLatLong`
		`(in)city: string`
		`(out)Zip: string`
			100	439

S3.1	`USWeather`	`GetWeatherReport`
		`(in)ZipCode: string`
		`(out)WeatherReport: string`
			85	384

S3.23	`Weather`	`GetCityForecastByZIP`
		`(in)ZIP: string`
		`(out)ForecastReturn: string`
			100	237

Tableau 5.6 — Informations sur les services sélectionnés qui requièrent le minimum d'adaptation et qui offrent le maximum de QdS.

illustrer cette transformation, nous présentons dans la Figure 5.9 le modèle MIS de la carte d'affinement de la section <*Réserver un hôtel, Arrêter, Par paiement*> (voir Figure 5.3) et le modèle BPEL qui lui correspond.

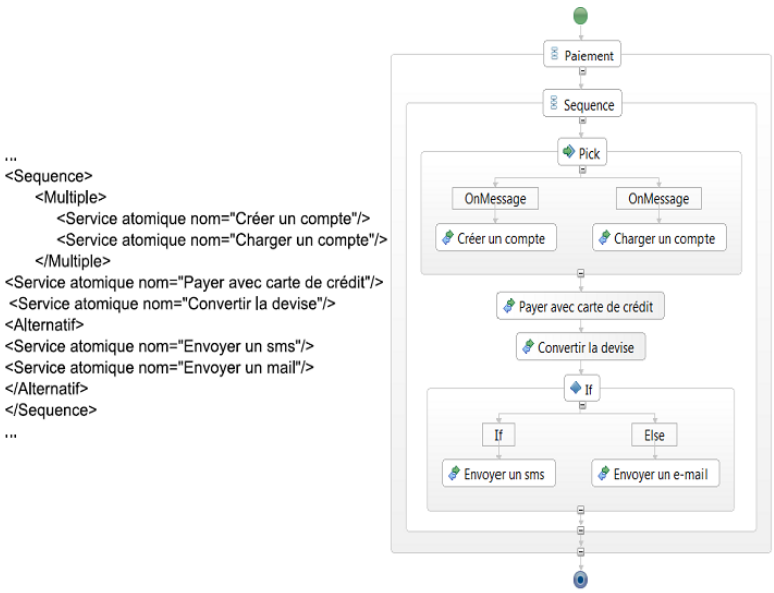


Figure 5.9 — Le modèle MIS de la carte d'affinement de la section <*Réserver un hôtel, Arrêter, Par paiement*> et le modèle BPEL qui lui correspond.

Le modèle MIS décrit une séquence de services qui est transformée vers un bloc *sequence* en BPEL. Les deux services $S_{Créer\ un\ compte}$ et $S_{Charger\ un\ compte}$ forment un point de variation Multiple transformé vers un bloc *pick*. Ces deux services sont suivis par $S_{Payer\ avec\ carte\ de\ crédit}$ et $S_{Convertir\ la\ devise}$. Finalement, nous trouvons $S_{Envoyer\ un\ sms}$ et $S_{Envoyer\ un\ e-mail}$ qui forment un point de variation Alternatif transformé vers un bloc *if else*. Le modèle BPEL élaboré est intentionnel. Pour pouvoir l'exécuter, il faudrait faire correspondre un service logiciel à chaque service intentionnel. Par exemple, pour le service $S_{Convertir\ la\ devise}$, nous pouvons exécuter l'un des services `CurrencyServer`, `CurrencyService` et `ForeignExchangeRates` obtenus par application de la FCA lors de l'étape de sélection de notre approche.

5.5 Validation

Nous proposons deux validations : une validation empirique pour les exigences fonctionnelles et une autre automatique pour les exigences non-fonctionnelles.

5.5.1 Validation des exigences fonctionnelles

Nous validons empiriquement les exigences fonctionnelles de ce cas d'étude en utilisant les deux mesures : la précision et le rappel.

Dans le Tableau 5.7, nous donnons les résultats expérimentaux des 39 services intentionnels de notre application d'arrangement de conférences.

La première colonne de ce tableau correspond aux services intentionnels. Dans la seconde colonne, nous avons, premièrement, la liste des mots-clés utilisés pour interroger *Service-Finder*, puis, le nombre de services opérationnels retournés par *Service-Finder* et enfin le nombre de services identifiés manuellement comme réels services pertinents et de haute QdS. Dans la dernière colonne, nous avons, premièrement, le nombre de services obtenus après l'étape de découverte de notre approche, deuxièmement, le nombre de services obtenus après l'étape de sélection et enfin le nombre de réels services pertinents et de haute QdS parmi ceux qui sont sélectionnés.

Par exemple, deux mots-clés sont utilisés pour rechercher des services opérationnels permettant de satisfaire le service intentionnel $S_{Envoyer\ un\ sms}$ qui sont : "Envoyer + SMS". La requête d'interrogation de *Service-Finder* retourne un ensemble de 162 services opérationnels. Parmi cet ensemble, seulement 17 services sont vérifiés manuellement comme de réels services pertinents et de haute QdS. L'étape de découverte réduit l'ensemble retourné par *Service-Finder* en un autre ensemble renfermant seulement 105 services pertinents. Parmi ces 105 services, seulement 18 services sont maintenus après l'étape de sélection. Nous avons vérifié manuellement que seulement 15 services parmi les 18 sont de réels services pertinents et de haute QdS.

Services intentionnels	*Service-Finder*			Notre approche		
	Mots-clés	Services retournés	Réels services pertinents et de haute QdS	Découverte	Sélection	Réels services pertinents et de haute QdS
$S_{Formuler\ une\ requête\ :\ ville\ départ/ville\ d'arrivée}$	'réserver+ vol+ ville'	11	2/11	6/11	2/6	2/2
$S_{Formuler\ une\ requête\ :\ compagnie\ aérienne}$	'réserver+ vol+ compagnie+ aérienne'	6	1/6	2/6	1/2	1/1
$S_{Formuler\ une\ requête\ :\ offre\ promotionnelle\ de\ vols}$	'réserver+ vol+ promotion'	3	1/3	1/3	1/1	1/1
$S_{Formuler\ une\ requête\ :\ date\ du\ vol}$	'réserver+ vol+ date'	12	2/12	5/12	2/5	2/2
$S_{Sélectionner\ le\ vol\ le\ moins\ cher}$	'trier+ vol+ prix'	6	1/6	3/6	1/3	1/1
$S_{Sélectionner\ la\ compagnie\ aérienne\ la\ plus\ réputée}$	'trier+ compagnie aérienne+	6	1/6	2/6	1/2	1/1
$S_{Sélectionner\ la\ destination\ la\ plus\ proche\ de\ l'hôtel}$	'trier+ vol+ destination'	6	1/6	1/6	1/1	1/1
$S_{Réserver\ un\ train}$	'train'	20	3/20	12/20	4/12	2/4
$S_{Réserver\ un\ taxi}$	'taxi'	9	2/9	4/9	2/4	2/2
$S_{Louer\ une\ voiture}$	'louer+ voiture'	6	1/6	4/6	1/4	1/1
$S_{Consulter\ les\ horaires\ des\ bus}$	'horaires+ bus'	1	1/1	1/1	1/1	1/1
$S_{Consulter\ les\ horaires\ de\ la\ navette}$	'horaires+ navette'	1	1/1	1/1	1/1	1/1
$S_{Formuler\ une\ requête\ :\ nom\ de\ l'hôtel}$	'réserver+ hôtel+ nom'	19	2/19	7/19	3/7	1/3
$S_{Formuler\ une\ requête\ :\ adresse\ de\ l'hôtel}$	'réserver+ hôtel+ adresse'	18	2/18	6/18	2/6	2/2
$S_{Formuler\ une\ requête\ :\ prix\ de\ l'hôtel}$	'réserver+ hôtel' prix'	18	2/18	7/18	2/7	2/2
$S_{Formuler\ une\ requête\ :type\ de\ l'hôtel}$	'réserver+ hôtel' type'	20	2/18	6/18	2/6	2/6
$S_{Sélectionner\ l'hôtel\ le\ moins\ cher}$	'trier+ 'hôtel+ prix'	11	1/11	6/11	1/6	1/1
$S_{Sélectionner\ l'hôtel\ le\ plus\ réputé}$	'trier+ hôtel+ type'	12	1/12	4/12	1/4	1/1
$S_{Sélectionner\ l'hôtel\ le\ plus\ proche}$	'trier+ hôtel+ localisation'	2	1/2	1/1	1/1	1/1

$S_{Télécharger\ le\ plan\ de\ la\ ville}$	'télécharger+ plan+ ville'	14	2/14	7/14	2/7	1/2
$S_{Consulter\ la\ météo}$	'météo'	112	25/112	46/112	25/112	25/25
$S_{Acheter\ un\ ticket\ de\ musée}$	'musée'	3	1/3	2/3	1/2	1/1
$S_{Acheter\ un\ ticket\ de\ parc\ d'attractions}$	'parc d'attractions'	0	-	-	-	-
$S_{Acheter\ un\ ticket\ de\ visite\ guidée}$	'visite guidée' visite guidée'	1	1/1	1/1	1/1	1/1
$S_{Formuler\ une\ requête\ :\ nom\ du\ restaurant}$	'restaurant+ nom'	6	1/6	4/6	2/4	1/2
$S_{Formuler\ une\ requête\ :\ spécialité\ du\ restaurant}$	'restaurant spécialité'	0	-	-	-	-
$S_{Formuler\ une\ requête\ :\ adresse\ du\ restaurant}$	'restaurant+ adresse'	1	1/1	1/1	1/1	1/1
$S_{Sélectionner\ le\ restaurant\ le\ moins\ cher}$	'trier+ restaurant+ prix'	2	1/2	2/2	1/2	1/1
$S_{Sélectionner\ le\ restaurant\ le\ plus\ réputé}$	'trier+ restaurant+ réputation'	2	1/2	2/2	1/2	1/1
$S_{Sélectionner\ le\ restaurant\ le\ plus\ proche}$	'trier+ restaurant+ localisation'	1	1/1	1/1	1/1	1/1
$S_{Annuler\ la\ réservation\ d'un\ vol}$	'annuler+ réservation+ vol'	10	1/10	7/10	1/7	1/1
$S_{Annuler\ la\ réservation\ d'un\ hôtel}$	'annuler+ réservation+ hôtel'	15	3/15	5/15	4/5	3/4
$S_{Annuler\ la\ réservation\ d'un\ restaurant}$	'annuler+ réservation+ restaurant'	1	1/1	1/1	1/1	1/1
$S_{Créer\ un\ compte}$	'créer + compte'	330	26/330	203/330	25/203	22/25
$S_{Charger\ un\ compte}$	'charger + compte'	29	4/29	20/29	5/20	4/5
$S_{Payer\ avec\ carte\ de\ crédit}$	'payer+ carte + crédit'	77	4/77	16/77	4/16	4/4
$S_{Convertir\ la\ devise}$	'convertir+ devise'	76	3/76	9/76	3/9	3/3
$S_{Envoyer\ un\ sms}$	'envoyer + sms'	162	17/162	105/162	18/105	15/18
$S_{Envoyer\ un\ e\text{-}mail}$	'envoyer + e-mail'	58	7/58	30/58	9/30	6/9

Tableau 5.7: Résultats des expérimentations réalisées sur l'application d'arrangement de conférences.

Services intentionnels	Précision	Rappel
$S_{Formuler\ une\ requête\ :\ ville\ départ/ville\ d'arrivée}$	2/2 (100%)	2/2 (100%)

$S_{\textit{Formuler une requête : compagnie aérienne}}$	1/1 (100%)	1/1 (100%)
$S_{\textit{Formuler une requête : offre promotionnelle de vols}}$	1/1 (100%)	1/1 (100%)
$S_{\textit{Formuler une requête : date/heure du vol}}$	2/2 (100%)	2/2 (100%)
$S_{\textit{Sélectionner le vol le moins cher}}$	1/1 (100%)	1/1 (100%)
$S_{\textit{Sélectionner la compagnie aérienne la plus réputée}}$	1/1 (100%)	1/1 (100%)
$S_{\textit{Sélectionner la destination la plus proche}}$	1/1 (100%)	1/1 (100%)
$S_{\textit{Réserver un train}}$	2/4 (50%)	2/3 (66.66%)
$S_{\textit{Réserver un taxi}}$	2/2 (100%)	2/2 (100%)
$S_{\textit{Louer une voiture}}$	1/1 (100%)	1/1 (100%)
$S_{\textit{Consulter les horaires des bus}}$	1/1 (100%)	1/1 (100%)
$S_{\textit{Consulter les horaires de la navette}}$	1/1 (100%)	1/1 (100%)
$S_{\textit{Formuler une requête : nom de l'hôtel}}$	1/3 (33.33%)	1/2 (50%)
$S_{\textit{Formuler une requête : adresse de l'hôtel}}$	2/2 (100%)	2/2 (100%)
$S_{\textit{Formuler une requête : prix de l'hôtel}}$	2/2 (100%)	2/2 (100%)
$S_{\textit{Formuler une requête : type de l'hôtel}}$	2/2 (100%)	2/2 (100%)
$S_{\textit{Sélectionner l'hôtel le moins cher}}$	1/1 (100%)	1/1 (100%)
$S_{\textit{Sélectionner l'hôtel le plus réputé}}$	1/1 (100%)	1/1 (100%)
$S_{\textit{Sélectionner l'hôtel le plus proche}}$	1/1 (100%)	1/1 (100%)
$S_{\textit{Télécharger le plan de la ville}}$	1/2 (50%)	1/2 (50%)
$S_{\textit{Consulter la météo}}$	25/25 (100%)	25/25 (100%)
$S_{\textit{Acheter un ticket de musée}}$	1/1 (100%)	1/1 (100%)
$S_{\textit{Acheter un ticket de visite guidée}}$	1/1 (100%)	1/1 (100%)
$S_{\textit{Formuler une requête : spécialité du restaurant}}$	1/2 (50%)	1/1 (100%)
$S_{\textit{Formuler une requête : nom du restaurant}}$	1/2 (50%)	1/1 (100%)
$S_{\textit{Formuler une requête : adresse du restaurant}}$	1/1 (100%)	1/1 (100%)
$S_{\textit{Sélectionner le restaurant le moins cher}}$	1/1 (100%)	1/1 (100%)
$S_{\textit{Sélectionner le restaurant le plus réputé}}$	1/1 (100%)	1/1 (100%)
$S_{\textit{Sélectionner le restaurant le plus proche}}$	1/1 (100%)	1/1 (100%)
$S_{\textit{Annuler la réservation d'un vol}}$	1/1 (100%)	1/1 (100%)
$S_{\textit{Annuler la réservation d'un hôtel}}$	3/4 (75%)	3/3 (100%)

$S_{\textit{Annuler la réservation d'un restaurant}}$	1/1 (100%)	1/1 (100%)
$S_{\textit{Créer un compte}}$	22/25 (88%)	22/26 (84.62%)
$S_{\textit{Charger un compte}}$	4/5 (80%)	4/4 (100%)
$S_{\textit{Payer avec carte de crédit}}$	4/4 (100%)	4/4 (100%)
$S_{\textit{Convertir la devise}}$	3/3 (100%)	3/3 (100%)
$S_{\textit{Envoyer sms}}$	15/18 (83.33%)	15/17 (88.24%)
$S_{\textit{Envoyer e-mail}}$	6/9 (66.67%)	6/7 (85.71%)
Average	91%	95.14%

Tableau 5.8: Précision et rappel de notre approche appliquée sur le cas d'étude de l'application d'arrangement de conférences.

Dans le Tableau 5.8, nous donnons la précision et le rappel correspondant aux 36 services intentionnels de l'application d'arrangement de conférences (nous ne considérons pas $S_{\textit{Acheter un ticket de parc d'attractions}}$ et $S_{\textit{Formuler une requête: spécialité du restaurant}}$ puisque nous n'avons pas trouvé de services qui leur correspondent). La précision de sélection de notre approche pour le service intentionnel $S_{\textit{Envoyer un sms}}$ est de 83,33% et le rappel est de 88,24%. Le Tableau 5.5 montre que la précision et le rappel de notre approche sont très élevés. La moyenne de la précision est de 91% et la moyenne du rappel est de 95,14% pour le cas d'étude de l'application d'arrangement de conférences. Le rappel est légèrement supérieur à la précision, cela signifie que notre approche permet de retourner, dans la plupart du temps, tous les services pertinents et de haute QdS. La précision est faible lorsqu'il s'agit de sélectionner à partir d'un ensemble restreint de services. C'est le cas par exemple de $S_{\textit{Formuler une requête: nom de l'hôtel}}$ où il fallait sélectionner à partir d'un ensemble formé de 7 services seulement (voir Tableau 5.5). Notre approche a retourné 3 services pertinents et de haute QdS alors qu'il en ait réellement qu'un seul service qui répond aux exigences fonctionnelles des utilisateurs et qui offre une haute QdS. La pertinence pour cet exemple est de 33,33% seulement. Ceci est dû essentiellement au mécanisme d'échelonnage de QdS qui donne de meilleurs résultats que lorsqu'il s'agit d'échelonner un très grand nombre de valeurs.

5.5.2 Validation des exigences non-fonctionnelles

La validation des exigences non-fonctionnelles est réalisée en utilisant la technique de simulation à évènements-discrets. Nous considérons deux chemins de notre cas d'étude à comparer : C1=•(ab1, bb1, bc3, cg1) et C2=•(ab2, bc1, cg1). Ces deux chemins permettent de satisfaire les mêmes intentions à savoir la réservation d'un hôtel et l'arrangement du transport vers cet hôtel. Par la simulation, nous proposons d'évaluer les deux propriétés de QdS : le temps de réponse et la disponibilité afin d'en

déduire quel chemin satisfait au mieux ces deux propriétés. L'évaluation de ces deux propriétés est faite localement pour chaque service, un modèle mathématique est par la suite utilisé pour calculer les valeurs globales. La simulation est réalisée en utilisant le simulateur réseau NS-2 [6]. Une série de 50 tests de simulation ont été réalisés et les valeurs moyennes de QdS ont été considérées. Le contexte d'exécution de ces tests est décrit par les éléments suivants :

- Equipement : PC doté d'un processeur Intel Core 2 Duo 2.80 GHz et une mémoire vive de 4.00Go.
- Connexion internet : Connexion NUMERIS avec un débit de 1Go/s.
- Topologie réseau : l'utilisateur est connecté à un fournisseur internet qui à son tour est connecté au serveur via un routeur :
  - Lien utilisateur-fournisseur internet : débit 1Go/s et un délai : 50ms.
  - Lien fournisseur internet-routeur : débit 2Go/s et un délai de 25ms.
  - Lien routeur-serveur : débit 2Go/s et un délai de 25ms.

La Figure 5.10 présente les résultats des tests de simulation réalisés. Ces résultats montrent que le chemin C2 est le plus approprié puisqu'il offre un temps de réponse moins élevé et une disponibilité plus élevée que le chemin C1.

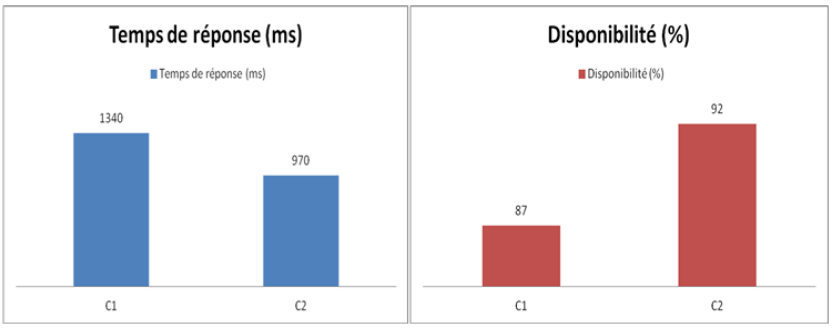


Figure 5.10 — Résultats de simulation des chemins C1 et C2

5.6 Conclusion

Nous avons expérimenté notre approche sur un cas d'étude d'une application 'arrangement de conférences formée de 39 services atomiques. Nous avons validé notre

6. http ://isi.edu/nsnam/ns/

approche empiriquement en termes de précision et de rappel et automatiquement en utilisant la technique de simulation. Les résultats des expérimentations empiriques ont montré que notre approche permet de découvrir, sélectionner et coordonner des services pertinents et de haute QdS avec une haute précision (91%) et une grande efficacité (95,14%). Les expérimentations automatiques ont pu permettre d'identifier les politiques de composition qui satisfait le mieux les exigences non-fonctionnelles.

Troisième partie

Conclusion et perspectives

Chapitre 6

Conclusion

La principale motivation qui a régit ce travail est la proposition d'une approche multi-perspective centrée exigences pour la composition de services Web. Cette approche place la description des services à composer sous une perspective centrée exigences et leur découverte, sélection, coordination et exécution sous une perspective centrée fonctions. Au niveau de la perspective centrée exigences, la conceptualisation des applications à base de services est exprimée en termes des exigences qu'elles permettent de satisfaire et non pas en termes de fonctions qu'elles permettent d'exécuter. Ceci évite les discordances conceptuelles entre les exigences et les services qu'ils les réalisent, garantie un haut niveau d'abstraction, et renforce la variabilité et la réutilisation de ces applications. Pour assurer le passage entre la perspective centrée exigences et la perspective centrée fonctions, un processus d'alignement des services aux exigences est mis en place. Ce processus consiste à découvrir, sélectionner, coordonner et exécuter les services logiciels qui répondent le mieux aux exigences fonctionnelles et non-fonctionnelles.
Notre approche consiste en un processus linéaire et itératif composé de cinq étapes successives réparties sur les deux perspectives.
La perspective centrée exigences inclut la première étape de l'approche qui assure la modélisation centrée exigences des services. Cette étape considère les services à composer comme des interactions métiers et elle les représente et les spécifie en termes des exigences qu'ils permettent de satisfaire. La modélisation est réalisée en utilisant le formalisme d'ingénierie des exigences la Carte qui représente les exigences par une séries de graphes composés d'intentions et de stratégies. La spécification des services métiers présentés par les cartes est faite en utilisant un modèle intentionnel de services appelé MIS. Ce modèle permet la description des services selon quatre facettes : l'interface, le comportement, la composition et la QdS. La perspective centrée fonctions inclut les quatre dernières étapes de l'approche. C'est la perspective où les services logiciels qui réalisent les exigences modélisées sont découverts, sélectionnés et coordonnées.
La deuxième étape du processus consiste à découvrir les services logiciels pertinents qui répondent le mieux aux exigences fonctionnelles. La découverte est faite par interrogation d'un moteur de recherche de services Web en utilisant des mots-clés extraits à partir des spécifications intentionnels des services métiers. L'ensemble des

services retournés par le moteur de recherche est réduit en procédant par un filtrage multi-niveaux : de QdS, syntaxique et sémantique.
La troisième étape consiste à sélectionner les services logiciels pertinents et de haute QdS. Ceci est réalisé par application de l'Analyse Formelle de Concepts AFC. L'AFC est un cadre de travail formel qui permet de grouper des individus ayant des propriétés communes et les organiser dans des structures ordonnées appelées treillis de concepts. L'AFC permet d'automatiser la tâche de sélection et ceci en fournissant une vue claire et organisée permettant d'identifier plus facilement les services pertinents et de haute QdS et leurs éventuels substituants. Pour la sélection des services composites, une variante de l'AFC s'ajoute qui est l'Analyse Relationnelle de Concepts ARC. L'ARC permet de raisonner sur la relation de composabilité entre services et sert alors d'identifier les services composables qui requièrent le minimum d'adaptation de composition et qui offrent le maximum de QdS.
La quatrième étape du processus consiste à élaborer le processus de coordination des services sélectionnés permettant leur exécution. Ceci est fait en utilisant la technique de transformation de modèles qui permet de générer automatiquement à partir d'un modèle intentionnel d'une application à base de services son processus d'orchestration BPEL.
Finalement, l'étape de validation consiste à vérifier si la composition élaborée satisfait ou non les exigences fonctionnelles et non-fonctionnelles. La validation des exigences fonctionnelles est réalisée empiriquement en utilisant deux mesures empruntées du domaine de recherche d'information : la précision et le rappel. La validation des exigences non-fonctionnelles est faite automatiquement en appliquant la technique de simulation à évènement discrets. Dans le cas où les résultats de validation obtenus montrent que les services composés ne répondent pas aux exigences des utilisateurs, des boucles de retour au niveau des trois premières étapes sont alors proposées. Les contributions apportées dans le cadre de ce travail de thèse sont :

- La définition et la caractérisation de deux perspectives pour la composition de services Web : une perspective centrée exigences appelée perspective "intentionnelle" qui intègre la variabilité essentielle au niveau de la représentation de la composition et de son modèle de spécification et une perspective centrée fonctions appelée perspective "opérationnelle" qui intègre la variabilité technique au niveau de la coordination et de l'exécution des services logiciels.

- L'extension du modèle intentionnel de services MIS par une couche de qualité de service QdS.

- La proposition d'une méthode de découverte automatique des services logiciels pertinents.

- La définition et la mise en œuvre d'une méthode de guidage automatique pour la sélection des services logiciels pertinents et de haute QoS. Cette méthode est basée sur deux cadres de travail formels qui sont l'Analyse Formelle de Concepts

AFC et l'Analyse Relationnelle de Concepts ARC.

– La proposition et l'implémentation d'un mécanisme de génération automatique des procédés de coordination des services logiciels se basant sur la technique de transformation de modèles.

– La validation de la composition : empiriquement en termes de précision et de rappel et automatiquement en utilisant la simulation à évènements discrets.

Chapitre 7

Perspectives

En termes de perspectives, nous proposons, à court terme, d'améliorer notre méthode de découverte et de sélection des services logiciels afin d'obtenir une meilleure précision. Ensuite, à moyen terme, nous planifions un certains nombres d'extensions pour l'amélioration de notre approche. Enfin, pour le long terme, nous suggérons quelques pistes exploratoires.

7.1 Amélioration de la méthode de découverte et de sélection des services logiciels

Pour la découverte et la sélection des services, nous envisageons quatre types d'améliorations. Tout d'abord, nous comptons utiliser la base lexicale WordNet afin d'élargir la liste des mots-clés utilisés pour la découverte des services logiciels. Ensuite, nous planifions de prendre en compte des propriétés de qualité autre que le temps de réponse et la disponibilité telles que la sécurité et la qualité d'expérience. Ceci permettra de raisonner sur plusieurs propriétés concurrentes ayant des priorités différentes. La sélection consistera alors à chercher les services offrant le meilleur compromis entre ces propriétés. Aussi, nous comptons considérer des niveaux de composabilité sémantiques en plus des niveaux de composabilité syntaxiques afin d'améliorer la sélection des services composites. Finalement, nous prévoyons de définir et d'appliquer des règles d'association au niveau des treillis générés par AFC e ARC afin d'identifier les différents types de corrélation qui peuvent exister entre les services tels que la substituabilité.

7.2 Extensions

A moyen termes, nous comptons apporter trois extensions au niveau de notre approche. Tout d'abord, nous visons de proposer un alignement mixte qui combine l'alignement bottom-up : des services aux exigences et l'alignement top-down : des exigences aux services. Ceci permettra de renforcer la concordance entre les exigences et les services logiciels qui les réalisent. Ensuite, nous prévoyons la définition et la mise en place de stratégies de rétroactions dans le cas où la validation montre que les services

composés ne répondent pas aux exigences modélisées. Ces stratégies s'appliqueront aux exigences et/ou aux services pour garantir la satisfaction des utilisateurs. Enfin, nous comptons renforcer la variabilité d'exécution des services par le pilotage des reconfigurations dynamiques. Ceci garantira l'auto-adaptation dynamique des services et la maîtrise de la complexité de leur évolution.

7.3 Pistes exploratoires

A long terme, nous proposons l'exploration de trois pistes incluant l'étude d'autres catégories de services et l'utilisation de nouvelles représentations de conceptualisation et de nouvelles techniques d'opérationnalisation d'applications à base de services. Tout d'abord, nous projetons de tester l'approche proposée sur des services Web sémantiques pour voir l'impact de leurs annotations sur la découverte et la sélection. Ensuite, nous comptons enrichir notre modèle conceptuel par l'utilisation des ontologies afin de bien spécifier les services métiers et leur composition. Finalement, nous planifions d'exploiter la technique de tissage de modèles au runtime pour piloter l'auto-adaptation dynamique des services et pour renforcer leur variabilité lors de l'exécution.

Publications

Les publications qui ont trait à ce travail sont présentées dans la liste ci-dessous. Une liste complète et mise à jour de ces publications est également disponible à l'adresse suivante `http://sites.google.com/site/mahadrisspageperso/research`.

Articles publiés dans des revues internationales avec arbitrage

1. Maha Driss, Yassine Jamoussi, Naouel Moha, Jean-Marc Jézéquel, and Henda Hajjami Ben Ghézala. *Une Approche Centrée Exigences pour la Composition de Services Web.* Revue d'Ingénierie des Systèmes d'Information (ISI),vol. 16, no. 2, pp. 97-125, 2011.
2. Yassine Jamoussi, Maha Driss, Jean-Marc Jézéquel, and Henda Hajjami Ben Ghézala. *QoS Assurance for Service-Based Applications Using Discrete-Event Simulation.* International Journal of Computer Science Issues (IJCSI),vol. 7, no. 4, pp. 1-11, 2010.

Chapitres de livres

1. Andreas Metzger, Salima Benbernou, Manuel Carro, Maha Driss, Gabor Kecskemeti, Raman Kazhamiakin, Kyriakos Krytikos, Andrea Mocci, Elisabetta Di Nitto, Branimir Wetzstein, and Fabrizio Silvestri. *Analytical Quality Assurance. Service Research Challenges and Solutions for the Future Internet.* Editors : Mike P. Papazoglou, Klaus Pohl, Michael Parkin, and Andreas Metzger, Springer-LNCS 6500, ISBN : 978-3-642-17598-5, pp.209-270, 2010.

Articles publiés dans des conférences internationales avec arbitrage

1. Maha Driss, Yassine Jamoussi, Jean-Marc Jézéquel, and Henda Hajjami Ben Ghézala.*A Multi-Perspective Approach for Web Service Composition.* iiWAS 2011, The 13th International Conference on Information Integration and Web-based Applications and Services, December 2011, ACM.
2. Zeina Azmeh, Maha Driss, Fady Hamoui, Marianne Huchard, Naouel Moha, and Chouki Tibermacine. *Selection of Composable Web Services Driven by User Requirements.* ICWS 2011, The 9th IEEE International Conference on Web Services, pp. 395-402, July 2011, IEEE Computer Society.
3. Maha Driss, Naouel Moha, Yassine Jamoussi , Jean-Marc Jézéquel, and Henda Hajjami Ben Ghézala. *A Requirements-Centric Approach to Web Services Modeling, Discovery, and Selection.* ICSOC 2010, The 8th International Conference On Service Oriented Computing, pp. 258-272, December 2010, Springer Verlag.

4. Maha Driss, Yassine Jamoussi, Jean-Marc Jézéquel, and Henda Hajjami Ben Ghézala. *A Discrete-Events Simulation Approach for Evaluation of Service-Based Applications.* ECOWS 2008, The 6th IEEE European Conference on Web Services, pp. 73-78, November 2008, IEEE Computer Society.

Articles publiés dans des ateliers

1. Maha Driss, Yassine Jamoussi and Henda Hajjami Ben Ghézala. QOS Testing of Service-Based Applications. IDT 2008, 3rd IEEE International Design and Test Workshop, pp.45-50, December, 2008, IEEE Computer Society.

Bibliographie

[1] M. Aiello and P. Giorgini, *Applying the tropos methodology for analysing web services requirements and reasoning about qualities of services*, International Journal of Web and Grid Services **5** (2004), no. 4, 20–26.

[2] G. Alonso, V. Machiraju, F. Casati, and H. Kuno, *Web services : concepts, architectures and applications*, Springer-Verlag, 2004.

[3] I-B. Arpinar, B. Aleman-Meza, R. Zhang, and A. Maduko, *Ontology-driven web services composition platform*, Proceedings of the 9th international conference on e-commerce technology (cec'04), 2004, pp. 46–152.

[4] L. Aversano, M. Bruno, G. Canfora, M. Di Penta, and D. Distante, *Using concept lattices to support service selection*, international Journal of Web Services Research (IJWSR) **3** (2006), no. 4, 32–51.

[5] Z. Azmeh, M. Driss, F. Hamoui, M. Huchard, N. Moha, and C. Tibermacine, *Selection of composable web services driven by user requirements*, Proceedings of the 9th international conference on web services (icws'11), 2011, pp. 395–402.

[6] Z. Azmeh, M. Huchard, C. Tibermacine, C. Urtado, and S. Vauttier, *Wspab : A tool for automatic classification and selection of web services using formal concept analysis*, Proceedings of the 6th european conference on web services (ecows'08), 2008, pp. 31–40.

[7] M. Barbut and B. Monjardet, *Ordre et classification : Algèbre et combinatoire n°2*, Hachette Université, 1970.

[8] S. Bennasri, *Une approche intentionnelle de représentation et de réalisation de la variabilité dans un système logiciel*, Ph.D. Thesis, 2005.

[9] D. Berardi, D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Mecella, *Automatic composition of e-services that export their behavior*, Proceedings of the 1st international conference on service oriented computing (icsoc'03), 2003, pp. 43–58.

[10] G. Birkhoff, *Lattice theory*, American Mathematical Society Colloquium Publications **25** (1950), no. 2, 204–206.

[11] K. Bontcheva, E.D. Valle, M. Erdmann, H. Lausen, N. Steinmetz, and A. Turati, *Realizing servicefinder : Web service discovery at web scale*, Proceedings of the 2nd european semantic technology conference (estc'08), 2008.

[12] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini, *Tropos : An agent-oriented software development methodology*, Journal of Autonomous Agents and Multi-Agent Systems **8** (2004), no. 3, 203–236.

[13] J. Cardoso, J. Miller, A. Sheth, and J. Arnold, *Modeling quality of service for workflows and web service processes*, Journal of Web Semantics **1** (2004), no. 3, 281–308.

[14] J.M. Chambers, W.S. Cleveland, B. Kleiner, and P.A. Tukey, *Graphical methods for data analysis*, Wadsworth and Brooks/Cole, 1983.

[15] S. Chollet, V. Lestideau, P. Lalanda, P. Colomb, and D. Moreno, *Heterogeneous service selection based on formal concept analysis*, Proceedings of the 6th world congress on services (services'10), 2010, pp. 367–374.

[16] Workflow Management Coalition, *Xpdl support and resources*, 2008. http ://www.wfmc.org/xpdl.html. Dernière consultation : 25 Octobre 2011.

[17] W.W. Cohen, P.D. Ravikumar, and S.E. Fienberg, *A comparison of string distance metrics for name-matching tasks*, Proceedings of the 3rd international workshop on information integration on the web (iiweb'03), 2003, pp. 73–78.

[18] W.W. Cohen and J. Richman, *Learning to match and cluster large high-dimensional data sets for data integration*, Proceedings of the 8th sigkdd international conference on knowledge discovery and data mining (sigkdd'02), 2002, pp. 475–480.

[19] M. Cremene, J-Y. Tigli, S. Lavirotte, F-C. Pop, M. Riveill, and G. Rey, *Service composition based on natural language requests*, Proceedings of the 7th international conference on services computing (scc'09), 2009, pp. 486–489.

[20] A. Dardenne, A. Van Lamsweerde, and S. Fickas, *Goal-directed requirements acquisition*, Science of Computer Programming **20** (1993), no. 1-2, 3–50.

[21] Developpez.com, *Soap : Communiquer avec un web service étendu*, 2010. `http://mbaron.developpez.com/soa/soap/`. Dernière consultation : 25 Octobre 2011.

[22] G. Diaz, M. E. Cambronero, M. Llanos Tobarra, V. Valero, and F. Cuartero, *Analysis and verification of time requirements applied to the web services composition*, Proceedings of the 3rd international workshop on web services and formal methods (ws-fm'06), 2006, pp. 178–192.

[23] J.A. Incera Diéguez, *Contributions à la modélisation et à la simulation accélérée de réseaux de communication*, Ph.D. Thesis, 2001.

[24] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, *Similarity search for web services*, Proceedings of the 30th international conference on very large data bases (vldb'04), 2004, pp. 372–383.

[25] M. Driss, Y. Jamoussi, and H. Hajjami Ben Ghézala, *Qos testing of service-based applications*, Proceedings of the 3rd international design and test workshop (idt'08), 2008, pp. 45–50.

[26] M. Driss, Y. Jamoussi, J.M. Jézéquel, and H. Hajjami Ben Ghézala, *A discrete-events simulation approach for evaluation of service-based applications*, Proceedings of the 6th european conference on web services (ecows'08), 2008, pp. 73–78.

[27] M. Driss, Y. Jamoussi, N. Moha, J.M. Jézéquel, and H. Hajjami Ben Ghézala, *Une approche centrée exigences pour la composition de services web*, Revue d'Ingénierie des Systèmes d'Information **16** (2011), no. 2, 97–125.

[28] M. Driss, N. Moha, Y. Jamoussi, J.M. Jézéquel, and H. Hajjami Ben Ghézala, *A requirements-centric approach to web services modeling, discovery, and selection*, Proceedings of the 8th international conference on service oriented computing (icsoc'10), 2010, pp. 258–272.

[29] S. Dustdar and W. Schreiner, *A survey on web services composition*, International Journal of Web and Grid Services **1** (2005), no. 1, 1–30.

[30] W. Emmerich and N. Kaveh, *Component technologies : Java beans, com, corba, rmi, ejb and the corba component model*, SIGSOFT Software Engineering Notes **26** (2001), no. 5, 311–312.

[31] T. Erl, *Soa principles of service design*, Prentice Hall, 2007.

[32] A. Etien, *Ingénierie de l'alignement : Concepts, modèles et processus*, Ph.D. Thesis, 2006.

[33] G. Fenza and S. Senatore, *Friendly web services selection exploiting fuzzy formal concept analysis*, Soft Comput. **14** (2010), no. 8, 811–819.

[34] H. Foster, S. Uchitel, J. Magee, and J. Kramer, *Compatibility verification for web service choreography*, Proceedings of the 2nd international conference on web services (icws'04), 2004, pp. 738–741.

[35] W. B. Frakes and R. Baeza-Yates, *Information retrieval : Data structures and algorithms*, Prentice-Hall, 1992.

[36] B. Ganter and R. Wille, *Formal concept analysis : Mathematical foundations*, Springer-Verlag New York, Inc, 1999.

[37] G. Gardarin, *Xml, des bases de données aux services web*, Dunod, 2007.

[38] J. Gortmaker, M. Janssen, and R. W. Wagenaar, *The advantages of web service orchestration in perspective*, Proceedings of the 6th international conference on electronic commerce (icec'04), 2004, pp. 506–515.

[39] I. Jacobson M. Griss and P. Jonsson, *Software reuse : architecture, process and organization for business success*, ACM Press/Addison-Wesley, 1997.

[40] J.Van Gurp, J.Bosch, and M.Svahnberg, *On the notion of variability in software product lines*, Proceedings of the 2nd working ieee/ifip conference on software architecture (wicsa'01), 2001, pp. 45–54.

[41] A.R. Hacene, *Étude de 'analyse formelle dans les données relationelles - application à la restructuration des modèles structuraux uml*, Ph.D. Thesis, 2006.

[42] K. Halmans and K. Pohl, *Communicating the variability of a software product family to customers*, Software and System Modeling **2** (2003), no. 1, 15–36.

[43] R. Hamadi and B. Benatallah, *A petri net-based model for web service composition*, Proceedings of the 14th australasian database conference (adc'03), 2003, pp. 191–200.

[44] L. Hou, Z. Jin, and B. Wu, *Modeling and verifying web services driven by requirements : An ontology-based approach*, Science in China Series F : Information Sciences **49** (2006), no. 6, 792–820.

[45] M. Huchard, A. R. Hacene, C. Roume, and P. Valtchev, *Relational concept discovery in structured datasets*, Annals of Mathematics and Artificial Intelligence **49** (2007), no. 1-4, 39–76.

[46] M. N. Huhns and M. P. Singhl, *Service-oriented computing : Key concepts and principles*, IEEE Internet Computing **9** (2005), no. 1, 75–81.

[47] M. Jackson, *Software requirements and specifications - a lexicon of practice, principles and pejudices*, ACM Press, Addison-Wesley, 1995.

[48] M. A. Jaro, *Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida*, Journal of the American Statistical Association **84** (1989), no. 406, 414–420.

[49] R. S. Kaabi, *Une approche méthodologique pour la modélisation intentionnelle des services et leur opérationnalisation*, Ph.D. Thesis, 2007.

[50] H. Kadima, *Conception orientée objet guidée par les modèles*, Dunod, 2005.

[51] R. Kazhamiakin, M. Pistore, and M. Roveri, *Formal verification of requirements using spin : A case study on web services*, Proceedings of the 2nd international conference on software engineering and formal methods (sefm'04), 2004, pp. 406–415.

[52] D. Kitchin, A. Quark, W.R. Cook, and J. Misra, *The orc programming language*, Proceedings of the 11th international conference on formal techniques for distributed systems (fmoods/forte'09), 2009, pp. 1–25.

[53] J-J. Le and F. He, *Automatic web services composition based on reasoning petri net*, Proceedings of the 7th international conference on advanced language processing and web information technology (alpit'08), 2008, pp. 569–574.

[54] Y. M. Driss, Jamoussi, J-M. Jézéquel, and H. Hajjami Ben Ghézala, *A multi-perspective approach for web service composition*, Proceedings of the 13th international conference on information integration and web-based applications and services (iiwas'11), 2011.

[55] R. MacNaughton and H. Yamada, *Regular expressions and state graphs for automata*, IEEE Transactions on Electronic Computers **9** (1960), no. 1, 39–47.

[56] K. Mahbub and G. Spanoudakis, *Run-time monitoring of requirements for systems composed of web services : Initial implementation and evaluation experience*, Proceedings of the 3rd ieee international conference on web services (icws'05), 2005, pp. 257–265.

[57] N. K. Malhotra, *Etudes marketing avec spss*, Pearson France, 2007.

[58] J-D. Mckeen and H-A. Smith, *Making it happen : Critical issues in it management*, Wiley, 2003.

[59] M. Mecella, F. P. Presicce, and B. Pernici, *Modeling e-service orchestration through petri nets*, Proceedings of the 3rd international workshop on technologies for e-services (tes'02), 2002, pp. 38–47.

[60] I. Mirbel and P. Crescenzo, *Des besoins des utilisateurs à la recherche de services web : Une approche sémantique guidée par les intentions*, Ingénierie des Systèmes d'Information **15** (2010), no. 4, 89–112.

[61] OASIS, *Owl-s : semantic markup for web services*, 2004. `http://www.w3.org/Submission/OWL-S/`. Dernière consultation : 25 Octobre 2011.

[62] ———, *Uddi version 3.0.2 - uddi spec technical committee draft*, 2004. `http://www.uddi.org/pubs/uddi_v3.htm`. Dernière consultation : 25 Octobre 2011.

[63] ———, *Web services choreography description language version 1.0*, 2005. `http://www.w3.org/TR/ws-cdl-10/`. Dernière consultation : 25 Octobre 2011.

[64] ———, *Web services business process execution language version 2.0*, 2007. `http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html`. Dernière consultation : 25 Octobre 2011.

[65] OMG, *Business process management initiative*, 2008. http ://www.bpmi.org/. Dernière consultation : 25 Octobre 2011.

[66] M. P. Papazoglou, *Service-oriented computing : Concepts, characteristics and directions*, Proceedings of the 4th international conference on web information systems engineering (wise'03), 2003, pp. 3–12.

[67] ———, *Extending the service oriented architecture*, Business Integration Journal (2005), 18–21.

[68] ———, *Web services : Principles and technology*, Prentice-Hall, 2007.

[69] M. P. Papazoglou and W-J. Van Den Heuvel, *Service oriented architectures : Approaches, technologies and research issues*, IEEE Internet Computing **16** (2007), no. 3, 389–415.

[70] D. Peng, S. Huang, X. Wang, and A. Zhou, *Management and retrieval of web services based on formal concept analysis*, Proceedings of 5th international conference on computer and information technology (cit'05), 2005, pp. 269–6275.

[71] G. Pirró and N. Seco, *Design, implementation and evaluation of a new semantic similarity metric combining features and intrinsic information content*, Proceedings of the 7th international conference on ontologies, databases, and applications of semantics (odbase'08), 2008, pp. 1271–1288.

[72] M. Pistore, M. Roveri, and P. Busetta, *Requirements-driven verification of web services*, Proceedings of the 1st international workshop on web services and formal methods (wsfm'04), 2004, pp. 95–108.

[73] P. Plebani and B. Pernici, *Urbe : Web service retrieval based on similarity evaluation*, IEEE Transactions on Knowledge and Data Engineering **21** (2009), no. 11, 1629–16421.

[74] N. Prat, *Goal formalisation and classification for requirements engineering*, Proceedings of 3rd international workshop on requirements engineering : Foundations of software quality (refsq'97), 1997, pp. 145–156.

[75] M. Pérez, I. Sanz, R. Berlanga, and M-J. Aramburu, *Semi-automatic discovery of web services driven by user requirements*, Proceedings of the 21st international conference on database and expert systems applications (dexa'10), 2010, pp. 62–75.

[76] G. Regev and A. Wegmann, *Remaining fit : On the creation and maintenance of fit*, Proceedings of the 5th international workshops on business process modeling, development and support (bpmds'04), 2004, pp. 131–137.

[77] B-H. Reich and I. Benbasat, *Factors that influence the social dimension of alignment between business and information technology objectives*, MIS Quarterly **24** (2000), no. 1, 81–113.

[78] P. Resnik, *Using information content to evaluate semantic similarity in a taxonomy*, Proceedings of the 14th international joint conference on artificial intelligence (ijcai'95), 1995, pp. 448–453.

[79] N. Robinson, *A requirements monitoring framework for enterprise systems*, Requirements Engineering Journal **11** (2005), no. 1, 17–41.

[80] C. Rolland, R. S. Kaabi, and N. Kraeim, *On isoa : Intentional services oriented architecture*, Proceedings of the 9th international conference on advanced information systems engineering (caise'07), 2007, pp. 158–172.

[81] C. Rolland, M. Kirsch-Pinheiro, and C. Souveyet, *An intentional approach to service engineering*, IEEE Transactions on Services Computing **3** (2010), no. 4, 292–305.

[82] C. Rolland and N. Prakash, *Bridging the gap between organizational needs and erp functionality*, Requirements Engineering **5** (2000), no. 3, 180–193.

[83] M. Rouached and C. Godart, *Requirements-driven verification of wsbpel processes*, Proceedings of the 5th international conference on web services (icws'07), 2007, pp. 354–363.

[84] R.Wille, *Restructuring lattice theory : an approach based on hierarchies of concepts.*, Rival, i. (ed.) : Ordered sets, 1982, pp. 445–470.

[85] G. Salton and C. Buckley, *Term-weighting approaches in automatic text retrieval*, Information Processing and Management **24** (1988), no. 5, 513–523.

[86] S. Sanlaville, *Environnement de procédé extensible pour l'orchestration : Application aux services web*, Ph.D. Thesis, 2005.

[87] E. Sirin, B. Parsia, D. Wu, J. Hendler, and D.Nau, *Htn planning for web service composition using shop2*, Web Semantics : Science, Services and Agents on the World Wide Web **1** (2004), no. 4, 377–396.

[88] M. Stal, *Web services : Beyond component-based computing*, Communications of the ACM **45** (2002), no. 10, 71–76.

[89] C-A. Sun, R. Rossing, M. Sinnema, P. Bulanov, and M. Aiello, *Modeling and managing the variability of web service-based systems*, Journal of Systems and Software **83** (2010), no. 3, 502–516.

[90] H. Sun, X. Wang, B. Zhou, and P. Zou, *Advanced parallel processing technologies*, Springer Berlin / Heidelberg, 2003.

[91] L-H. Thevenet, *Proposition d'une modélisation conceptuelle d'alignement*, Ph.D. Thesis, 2009.

[92] W.M.P. van der Aalst and A.H.M. ter Hofstede, *Yawl : yet another workflow language*, Information Systems **30** (2005), no. 4, 245–275.

[93] B. Verlaine, Y. Dubois, I-J. Jureta, and S. Faulkner, *Towards automated lignment of web services to requirements*, Proceedings of the 1st international workshop on the web and requirements engineering (were'10), 2010, pp. 5–12.

[94] W3C, *Web service choreography interface (wsci) 1.0*, 2002. `http://www.w3.org/TR/wsci/`. Dernière consultation : 25 Octobre 2011.

[95] _____, *Web services architecture*, 2004. `http://www.w3.org/TR/ws-arch/`. Dernière consultation : 25 Octobre 2011.

[96] _____, *Web services glossary*, 2004. `http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/`. Dernière consultation : 25 Octobre 2011.

[97] _____, *Soap version 1.2 part 1 : messaging framework (second edition)*, 2007. `http://www.w3.org/TR/soap12-part1/`. Dernière consultation : 25 Octobre 2011.

[98] _____, *Web services description language (wsdl) version 2.0 part 1 : Core language*, 2007. `http://www.w3.org/TR/wsdl20/`. Dernière consultation : 25 Octobre 2011.

[99] J. Xiang, L. Liu, W. Qiao, and J. Yang, *Srem : A service requirements elicitation mechanism based on ontology*, Proceedings of the 31st annual international computer software and applications conference (compsac'07), 2007, pp. 196–203.

[100] J. Yang and M. P. Papazoglou, *Service components for managing the life-cycle of service compositions*, Information Systems **29** (2004), no. 2, 97–125.

[101] E. Yu, *Towards modelling and reasoning support for early-phase requirements engineering*, Proceedings of the 3rd symposium on requirements engineering (re'97), 1997, pp. 2444–2448.

[102] K. Zachos, N. Maiden, X. Zhu, and S. Jones, *Discovering web services to specify more complete system requirements*, Proceedings of the 19th international conference on advanced information systems engineering (caise'07), 2007, pp. 142–157.

[103] Y. Zhang, Z. Zheng, and M. R. Lyu, *Wsexpress : a qos-aware search engine for web services*, Proceedings of the 8th international conference on web services (icws'10), 2010, pp. 91–98.